



కంప్యూటర్ విద్య

10వ తరగతి



తెలంగాణ రాష్ట్ర ప్రభుత్వ ప్రచురణ,
హైదరాబాదు



రాష్ట్ర విద్యా పరిశోధన శిక్షణా సంస్థ,
తెలంగాణ రాష్ట్రం, హైదరాబాదు

ICT TEACHER HANDBOOK

(Computer Education)

Class - X



**State Council of Educational Research & Training,
Telangana, Hyderabad.**



Content Development Committee

Sri G.V.S.P. Kumar, Trustee, Remedia Trust, Hyderabad.

Sri M. Jagadish Babu, Programme Officer, C-DAC, Hyderabad.

Sri Mahesh Varal, Consultant, I.T., Hyderabad.

Sri M. Ramanjaneyulu, Lecturer, DIET Vikarabad, Rangareddy.

Coverpage Designing

Sri K. Sudhakara Chary, SGT, UPS Neelukurthy, Maripeda, Warangal.

Sri Kannaiah Dara, Computer Operator, SCERT, Hyderabad.

Editing & Co-ordination

Dr. P. Jani Reddy

Lecturer, DIET, Vikarabad
Rangareddy Dist.

Dr. N. Upender Reddy

Professor & Head, C & T Department,
SCERT, Telangana, Hyderabad.

Advisor

Sri G. Gopal Reddy

Former Director
SCERT,
Hyderabad.

Sri S. Jagannadha Reddy

Director
SCERT,
Telangana, Hyderabad.



Foreword

Information Communication Technology (ICT) is one of the rapidly changing area in domain of learning. The main purpose of introducing ICT in schools is to help the students to make use of different ICT tools in enhancing their learning. The ICT education helps the students to face the challenges of 21st century and attain the millennium developmental goals in education. ICT also helps in reducing the digital divide among the rural and urban students.

As per the UNESCO “technology can be a powerful education multiplier, but we must know how to use it. It is not enough to install technology into classrooms- it must be integrated into learning. Nothing can be substitute for a good teacher. It is not technology itself that empowers people- empowerment comes from skills and knowledge”.

The teachers have become more critical than ever- the challenge is how to enable teachers not only to overcome the technology barriers but also to empower them to integrate appropriate technology into the teaching and learning process.

This handbook contains the required content in using the different tools and use them in other subjects i.e. Mathematics, Science, Social Science and languages.

The SCERT, Telangana, Hyderabad appreciates the efforts of the content experts and other team members in developing this ICT handbook. We also acknowledge the C-DAC, Hyderabad for sharing their content related to software security in the preparation of this handbook.

We hope, the teacher may find this handbook useful and transact it in the classroom to enhance the learning levels of the students and also infuse confidence among them as they were not left behind in using technology on par with other students who are studying in corporate schools.

Director
SCERT,
Telangana, Hyderabad.

Class - X

CONTENTS

Sl.No.	Name of the Chapter	Page No.
	Introduction	1 - 4
1.	Advantages of ICT Education – Advantages	5 - 10
2.	More about Tablet PC	11 - 13
3.	More about Web Camera	14 - 16
4.	Understand Modem	17 - 20
5.	Factors Effecting Computer performance	21 - 22
6.	Configuring Windows for Network	23 - 28
7.	Computer Virus	29 - 30
8.	Windows – Files and Folders	31 - 39
9.	Animations in Power Point	40 - 44
10.	Creating Tables Word document	45 - 53
11.	Charts in Excel	54 - 59
12.	Understand Queries, Forms and Reports in MS Access	60 - 70
13.	Object Oriented Programming (OOPS)-Concept	71 - 123

Instructions to Teachers

The main purpose of introducing ICT in school syllabus is to help the children understand how to make use of ICT tools in enhancing their learning.

- This material is meant for teacher only. Hence, the teacher should read, understand and add their experience while transacting the same to students.
- Prepare a year plan in advance keeping in view of the number of periods available under computer education.
- Read all the topics given in the handbook and prepare thoroughly before start teaching.
- Refer to relevant material and consult the experts in case of any doubts.
- Before going to practice, the theory/ content should be explained to students then only demo and practice sessions should be taken up.
- All abbreviations must be thoroughly practised by the teacher before teaching.
- Update your knowledge in software and hardware in addition to the latest ICT tools. Use internet to update the knowledge in the given topics.
- Ensure all children get equal opportunities in practices.
- There will be two periods in a week for ICT (Computer Education) subject, accordingly, teacher has to plan for transaction.
- Few projects are given at the end of the handbook. All students should do these projects. In addition to these, you may consult your colleagues and identify some more projects and assign to your students.

Assessment :

Children's performance should be assessed at the end of each summative based on the indicators given below:

- Students are able to operate and use the ICT tools;
- Students are able to learn subject-wise content by using ICT tools.

There is no written test. The children performance should be assessed through observation based on their participation and their ability in using the ICT tools in their learning.

Assessment Indicators

In addition to grading the qualitative Indicators are to be written in the students' progress report under Computer Education. The following are the indicators:

- A1** Students can use all the tools of the ICT and can use them effectively in their learning.
- A2** Students can use all the tools of the ICT and able to use them in their learning.
- B1** Students are able to use different tools of ICT and able to use most of them in their learning.
- B2** Students are able to use the different tools of ICT and use some of them in their learning.
- C1** Students are able to use the ICT tools and able to use a few of them in their learning.
- C2** Students are able to use some of the ICT tools and try to use them in their learning.
- D1** Students are able to identify the ICT tools and try to use a few of them in their learning.
- D2** Students are able to identify some of the ICT tools but they are unable to use them in their learning.

Note: The computer education is linked with Work Education and made a single paper under other curricular subject areas. Hence, out of 50 marks, each 25 marks are to be allotted to Work Education and Computer Education.

Information and Communication Technology (ICT)

Introduction

The field of Information and Communication Technology (ICT) is evolving at such a pace, where concepts, technology and terminology are continuously changing. ICT helps to bridge the digital divide amongst students of various socio economic and other geographical barriers. Information and Communication Technology (ICT) is universally acknowledged as an important catalyst for social transformation and national progress.

Information and communication technology which is a by-product of science and technology explosion has revolutionized the world of learning. It is very essential to integrate IT with education in order to have the advantage of ICT education.

Around the world, policymakers and educators have high hopes for ICT in the classroom as a springboard to students “21st century skills”—that combines the competencies of problem solving, critical thinking and managing their own learning that is needed for success in the global workplace.

Information and Communication Technology (ICT) has dominated every walks of life affecting right from bus & railway reservations, hotel industry, online money transfers , bill payments, in class room teaching and learning process, distance education, e-learning and film making etc..

The learning activities through ICT make a difference. Students are much more likely to learn to solve real-world problems and collaborate productively with their peers, if their learning activities are carefully designed to offer opportunities.

The aim of 21st Century education is being redefined. It is not only for employment generation but also to create a better world through understanding and development of human qualities.

In this context, the Government of India has announced 2010 – 2020 as the decade of Innovation. For which, reasoning and critical thinking skills are essential. And these skills are to be inculcated at the school level for which ICT tools and techniques should be integrated into class room instructions right from the primary

education level, so that the children develop the required skills. In this matter the web is an open source and the child should know how to grab it.

According to UNESCO, education is important to achieve the Millennium Development goals; the following are reasons for which the ICT in education is a key aspect. i.e., *‘More people would grow and develop; More people would learn and know; More people would be equal and just; More children would survive and live; More mothers would be healthier; More people would be able to combat illness; More people would think of the future; More people would work together.’*

The purpose of this material is to create an awareness and practice among the student and teacher communities with a “diverse set of technological tools, resources used to communicate, create, disseminate, store, and manage information.” These technologies include computers, internet, broadcasting technologies (radio and television) and telephony.

The relevant and contemporary content, lucid language, attractive illustrations and constructive exercises make the learning of computers more meaningful and enriching.

The journey of the student starts right from the ICT tools available in and around such as Radio, Tape recorder, TV, Mobile and Computer. The students of primary level learn to draw by using MS & Tux Paint, learn to type small words, sentences, paragraphs, making documents by using typing tutor and MS Word. Moving away a little from these applications, the students can learn MS Excel for mathematical calculations & graphs and MS Power Point for making subject wise presentations. Further, the students learn the concepts of Data Base Management System (DBMS), Internet, Networking and computer languages, maintaining the computers and its peripherals. The students are also introduced to learn about the Antivirus, Computer security and privacy and open source software technologies.

This material is useful in acquiring the concepts in a better way which make the student journey in learning more fruitful and engrossing. It has been developed in tune with the guidelines given by NCF-2005 and with various activity based methods. This will lay down a path to create interest to take up high level computer education courses in future and career.

The government had initiated Computer Education and Computer Aided Learning in selected Primary, Upper Primary and High Schools. A huge number of teachers in the State already have been trained to use computers in their regular class room transactions and a lot of computer aided learning material also has been developed and supplied to schools.

As part of Education Technology Policy, the Education Department of Andhra Pradesh has been developing a large quantity of material in the form of Audio (Radio Programs under Vindam Nerchukundam), Audio and Video based programs (SIET and SAP Net) and Computer based programs(CDs) in collaboration with NGOs.

The Objectives of the ICT in School Education:

- To inculcate the ICT skills among the students of government schools.
- To bridge the digital divide between rural and urban students.
- To create computer awareness and literacy among students and teachers.
- To provide ICT environment in schools to make teaching-learning process effective and interesting
- To train the teachers on computer syllabi, emerging information and communication technologies.
- To develop confidence in students to use computers in future.
- Student-centered pedagogies that promote personalized and powerful learning for students;
- Extending learning beyond the classroom in ways most relevant to knowledge-building and problem-solving in today's world; and
- ICT integration into pedagogy in ways that support learning goals. It is important to note that ICT use is not a goal in itself, but a tool to broaden and deepen learning opportunities.

Teaching Learning Process

The ICT material has been developed from class 1 to class 10. The teacher can act as a facilitator to help the students in their learning process. In every Unit the teacher explains the concepts and students do the activities/projects with ICT tools and finally able to integrate with their subjects.

Academic Standards and Assessment

Academic Standards

- The students are shall operate and use the ICT tools;
- Students are shall learn subject wise contents through the ICT tools.

Assessment Process

At the end of the each chapter exercises and projects have been given. On the basis of the students performance appropriate grade may be awarded in summative assessment.

What is ICT – Advantages**Learning Objectives**

In this chapter you will be able to understand about ;

- ICT in Education
- Advantages of ICT

Scope of ICT in Rural Development

Recent developments in Information and Communication Technology (ICT) have introduced a plethora of opportunities for development in every conceivable area. ICT as an enabler has broken all bounds of cost, distance and time. The fusion of computing and communications, especially through the internet has reduced the world indeed into global village creating new actors and new environments.

One of the major components and driving force of rural development is communication. Conventionally, communication includes electronic media, human communication & now information technology (IT). All forms of communications have dominated the development scene in which its persuasive role has been most dominant within the democratic political frame work of the country. Persuasive communication for rural development has been given highest priority for bringing about desirable social and behavioral change among the most vulnerable rural poor and women. Initially, the approach lacked gender sensitivity and empathy of the communicators and development agents who came from urban elite homes. Added to these constraints is political will that still influences the pace and progress of rural development. Technological changes further compounded the direction of rural development as information and communication technology (ICT) has been thought by communication and development workers as a panacea for other ills that obstructs the development process. It has lead to indiscriminate applications and use of ICT in every aspect of information dissemination, management & governance of development. While there are few shining examples of achievements of ICT in development, there are a large number of failures and unauthenticated claims.

The closing decade of twentieth century was the opening of historic information and communication technology interventions for development. This period has witnessed enormous and unprecedented changes in every aspect of communications technologies policies, infrastructure development and services. The ICT boom in India has already started changing the lives of Indian masses. The role of ICT in Rural Development must be viewed in this changing scenario.

Expected Role of ICT in Rural Development

Since the dawn of independence, concerted efforts have been made to ameliorate the living standard of rural masses. So, rural development is an integrated concept of growth, and poverty elimination has been of paramount concern in all the five year plans. Rural Development (RD) programmes comprise of following:

Provision of basic infrastructure facilities in the rural areas e.g. schools, health facilities, roads, drinking water, electrification etc.

- Improving agricultural productivity in the rural areas.
- Provision of social services like health and education for socio-economic development.
- Implementing schemes for the promotion of rural industry increasing agriculture productivity, providing rural employment etc.
- Assistance to individual families and Self Help Groups (SHG) living below poverty line by providing productive resources through credit and subsidy.

Communication has been seen by a large number of development planners as a panacea for solving major social evils and problems. Apart from development, the introduction of communication in the educational process for open and distance learning is seen as step towards improving the quality of education and bridging the social and educational gap. ICT can be used towards betterment of education, agriculture, social awareness and health and hygiene.

Experiences and experiments

Communication has been seen by a large number of development planners as a panacea for solving major social ills and problems. Apart from development, the introduction of communication in the educational process for open and distance learning is seen as step towards improving the quality of education and bridging the social and educational gap.

However, experience indicates that those rich who could afford to have access to private resources have hogged the advantage whether development or education. In this respect it seems that communication technology has, in no way has helped the poor for improving their socio-economic condition. Primarily the responsibility of rural development remained with the government. In the pre-economic liberalization period, i.e. before 1992 broadcast media were used to reach the large rural population or target groups for the rural development projects. In the post economic liberalization period, rural development projects added information and communication technology (ICT) to provide individual need based information in broad development areas through Internet.

After independence, the government took upon itself the major responsibility of development. Hence, the central and state governments carried out development projects. Two such projects are briefly described.

Radio for Rural Development Popularly known as “Radio Farm Forum” was one of the earliest efforts in the use of radio for rural development. The experiment was carried out from February to April 1956 in five districts of Maharashtra State by All India Radio (AIR). Rural listener groups were organized, who would listen to radio broadcasts twice a week at 6.30p.m. for half an hour. “The group then stayed together for discussion of what they had heard, the discussion lasted usually, about half an hour, seldom less, frequently more”. The summative impact evaluation indicated positive outcome of radio rural forum. Impressive knowledge gains as a result of radio listening were reported across illiterates and literates, agriculturists and non-agriculturists, village leaders and others. However, over a period of time the project withered away.

Satellite Instructional Television Experiment (SITE) is considered to be one of the biggest techno-social communication experiments in education and rural development. The one-year experiment (August 1975 - July 1976) aimed to provide direct broadcasting of instructional and educational television in 2400 villages in states of Andhra Pradesh, Bihar, Karnataka, Madhya Pradesh, Orissa and Rajasthan. Over 500 conventional television sets spread over 335 villages in Kheda district, Gujarat was also part of SITE. Satellite technologists had called SITE as leap fogging from bullock cart stage to satellite communication, which did not discriminate between rural poor and urban rich for information and communication. It had given 50 years communication lead to rural poor of the country.

SITE provided telecast for rural primary school children in the age group 5 - 12 years studying in grades 1-5. Rural adults viewed television programmes on improved agricultural practices, health and family planning. They were also able to view news. Television was considered as window to the world. The telecast reliability was above 99 per cent during the experiment period. More than 90 per cent direct reception television sets were in working.

Both quantitative (survey) and qualitative in-depth (anthropological holistic study) evaluation indicated modest gains in some areas, whereas no gain or negative gain in other areas. The one-year duration was thought to be too little for any positive results. Based on the experiences and positive gains, INSAT satellite was launched in 1981. Since then a series of INSAT satellites have been launched and used for nationwide television telecast for education and development. The sad part is that, in spite of best efforts, satellite television has been used for entertainment more than rural development. I am sad that my prediction came true that satellite television will be used for entertainment and not rural development.

Communication Technology and Rural Development in India could not be operationalised for large-scale implementation in one form or the other. Lack of political will and indifference of bureaucracy killed the rural development project even before it could help poor to take advantage of radio broadcast.

ICT and e-Governance for Rural Development

Several states have initiated the creation of State Wide Area Networks (SWAN) to facilitate electronic access of the state and district administration services to the citizens in villages. The Information and Communication Technologies (ICT) are being increasingly used by the governments to deliver its services at the locations convenient to the citizens. The rural ICT applications attempt to offer the services of central agencies (like district administration, cooperative union, and state and central government departments) to the citizens at their village door steps. These applications utilize the ICT in offering improved and affordable connectivity and processing solutions.

Computerization of land records have been a great success in application of ICT in rural development. Land records are great importance to contemporary socio economic imperatives and their revision and updation are necessary for capturing the changes in rural social dynamics. Land records are an important part of rural development. The govt. of India started the centrally sponsored scheme of Computerization of Land Records (CoLR) in 1988-89 with main objectives of:

- Creating database of basic records
- Facilitating the issues of copies of records
- Reducing work load by elimination of drudgery of paper work
- Minimizing the possibilities manipulation of land records, and
- Creating a land management information system

The farmers were largely benefited CoLR. The farmers can get all necessary records when they need it, these records are free from human arbitrations, the updating becomes easy, free from harassment and the farmers had direct access to information regarding their property.

Challenges of application of ICT in Rural Development

ICTs alone can't bring about rural development. Education is one of the basic problem for application of ICT as 40% of India's population is illiterate. All modern economies have demonstrated in the past that education is the first step to building the capacity which people can then use. If the Indian economy grows at 5-6 per cent per annum as it has been growing over last 2-3 years, then over 10-15 years the size of the Indian economy would have doubled. Even with this level of growth it cannot by any means bridge disparities and eradicate poverty. Therefore introducing ICTs alone will not meet the development challenge. For ICTs to succeed in India, education for all must be the first priority.

It is, of course, important to note that the proportion of the economy involved in some or other form of adaptation or usage of ICT is still very small. The proportion of people involved in the ICT Industry, especially in the rural areas is negligible. Thus, another priority action, in order for the benefits of ICT to trickle down as well as contribute to the rural prosperity, would involve setting up several rural and village level micro-enterprises.

The basic challenges that usage of ICT for rural development faces are- Illiteracy amongst the vast multitude of people

Major power-cuts and 'brown-outs' affecting the country-side ranging from 5 to 12 hours every day. Even though uninterrupted power supply systems are used; yet they prove insufficient to cope up with the power breakdowns.

Serious band-width issues and connectivity problems. Even though technology is available to upgrade the band-width; not enough resources have been budgeted by the

Government to change this scenario. However once a few projects for the upgradation of the band-width on the anvil get commissioned, there should be a significant improvement in the connectivity.

Financing difficulties encountered by the local grass root level institutions as well as by the state governments. Drastic steps are needed to inject funds for the development of the ICTs in the rural areas; increasingly by the participation of the private sector.

Acute shortage of project leaders and guides who could ensure implementation of the ICTs at the grass root levels. Unfortunately most professionals want to work in the urban areas where there are ample opportunities available to them for growth as well as prosperity. In the absence of these ‘techno-catalytic’ resources; development of ICTs in the rural areas will always be very slow.

Learning Objectives

In this chapter you will be able to understand about ;

- More about Tablet PC.

In our earlier class we have understand various types of tablet pc, in this chapter let us understand the functionality and usage of tablet pen.

Using the tablet pen

One of the great things about a Tablet PC is that you can use the tablet pen to interact with items on the screen. Here are some tips for using your tablet pen:

Make sure your Tablet PC screen and pen are properly calibrated

Calibration ensures that your Tablet PC correctly recognizes the position of your pen. In Windows 7, when you calibrate for the first time, the screen uses 16 different reference points. This initial calibration applies to all users of the Tablet PC. If you calibrate again, only 4 points are used, and apply only to your user account.

Use your tablet pen even if you're using a convertible Tablet PC in laptop mode

You might find that this feels more natural than using a mouse or the pointing device on your keyboard.

Turn on flicks

Use flicks to perform actions quickly and easily with your tablet pen. You can use navigational flicks (drag up, drag down, move back, and move forward) and editing flicks (copy, paste, undo, and delete). You can also customize flicks for actions that you use often. For example, if you frequently use keyboard shortcuts, such as F5 or Ctrl+B, you can assign flicks to perform the same functions as pressing those key combinations.

Use Snipping Tool to capture an area of your screen

After snipping, you can use the captured image in an e mail message, a Windows Journal file, or anywhere else that you can paste an image. For example, use Snipping Tool

to highlight important information in a spreadsheet and send it in e mail. Or, use it to capture an interesting headline from a website to use in a presentation. You can also write on the captured image with your tablet pen.

Use check boxes to select multiple items

Check boxes are the easiest way to select multiple items, such as files and folders, with your tablet pen—just point to each item and select each item’s check box.

Making the most of Input Panel

With Tablet PC Input Panel, you can enter and correct text without using the keyboard. Here are some tips to make Tablet PC Input Panel easier to use:


- **Customize how you access Input Panel.** You can change where, when, and if Input Panel appears on your screen

Use gestures to delete, split, and join words that you write, and to control the insertion point. Gestures are motions that you make or symbols that you draw with your tablet pen

- **Change how you use the Insert button.** You can set the Insert button so that it works when you point to it with your pen. This makes entering text faster because, instead of tapping the button, you can just wave your pen over it.

Improving handwriting recognition accuracy

It can be frustrating if your Tablet PC doesn’t recognize your handwriting correctly. Here are some tips for improving the chances that your handwriting is recognized correctly:

- Provide lots of handwriting samples in Handwriting Personalization.
- Open Handwriting Personalization by tapping the **Start** button . In the search box, type **personalize handwriting recognition**, and then, in the list of results, tap **Personalize handwriting recognition**.
- Improve handwriting recognition for words such as proper names, acronyms, technical terms, and abbreviations by adding those words to the handwriting dictionary.

Using tablet buttons

- You can use the buttons on your Tablet PC (called **tablet buttons**) to perform common tasks quickly. For example, if you frequently open Windows Journal, you can set a tablet button to do this for you.
- Tablet buttons
- You can assign two actions to most tablet buttons. Press a button once quickly to perform one action. Press and hold the button to perform the other action.



Summary

In this chapter we have understand about the tablet PC pen tool.



Key Words

- Snipping
- Panel
- Convertible
- Calibrated.

Learning Objectives

In this chapter you will be able to understand about ;

- About Web Cameras

A webcam is a video camera that feeds its image in real time to a computer or computer network. Unlike an IP camera (which uses a direct connection using ethernet or Wi-Fi), a webcam is generally connected by a USB cable, FireWire cable, or similar cable.

Their most popular use is the establishment of video links, permitting computers to act as videophones or videoconference stations. The common use as a video camera for the World Wide Web gave the webcam its name. Other popular uses include security surveillance, computer vision, video broadcasting, and for recording social videos.

**Early development**

First developed in 1991, a webcam was pointed at the Trojan Room coffee pot in the Cambridge University Computer Science Department. The camera was finally switched off on August 22, 2001. The final image captured by the camera can still be viewed at its homepage. The oldest webcam still operating is FogCam at San Francisco State University, which has been running continuously since 1994

Later developments

One of the most widely reported-on webcam sites was JenniCam, created in 1996, which allowed Internet users to observe the life of its namesake constantly, in the same vein as the reality TV series *Big Brother*, launched four years later. Other cameras are mounted overlooking bridges, public squares, and other public places, their output made available on a public web page in accordance with the original concept of a “webcam”. Aggregator websites have also been created, providing thousands of live video streams or up-to-date still pictures, allowing users to find live video streams based on location or other criteria.

Uses

Childcare webcast video monitoring

Childcare webcams can offer improved security, communication, and increased service value in daycare facilities. According to researchers and industry leaders, as many as 100 childcare facilities add Internet viewing systems each month, and the total number of centers with some form of Internet monitoring runs into the thousands. In the United States, private services have been offering dedicated webcasting systems to centers nationwide for several years as of 2010.

Commerce

Webcams have been increasingly used for Augmented Reality experiences online. One such function has the webcam act as a ‘magic mirror’ to allow an online shopper to view a virtual item on themselves. The Webcam Social Shopper is one example of software that utilizes the webcam in this manner.

Videocalling and videoconferencing

As webcam capabilities have been added to instant messaging, text chat services such as AOL Instant Messenger, and VoIP services such as Skype, one-to-one live video communication over the Internet has now reached millions of mainstream PC users worldwide. Improved video quality has helped webcams encroach on traditional video conferencing systems. New features such as automatic lighting controls, real-time enhancements (retouching, wrinkle smoothing and vertical stretch), automatic face tracking and autofocus, assist users by providing substantial ease-of-use, further increasing the popularity of webcams.

Video security

Webcams are also used as security cameras. Software is available to allow PC-connected cameras to watch for movement and sound, recording both when they are detected. These recordings can then be saved to the computer, e-mailed, or uploaded to the Internet. In one well-publicised case, a computer e-mailed images of the burglar during the theft of the computer, enabling the owner to give police a clear picture of the burglar’s face even after the computer had been stolen.

Recently webcam privacy software has been introduced by such companies such as Stop Being Watched or Webcamlock. The software exposes access to a webcam and prompts the user to allow or deny access by showing what program is trying to access the webcam. Allowing the user to accept a trusted program the user recognizes or terminate the attempt immediately. Other companies on the market manufacture and sell sliding lens covers that allow users to retrofit the computer and close access to the camera lens.

In December 2011, Russia announced that 290,000 Webcams would be installed in 90,000 polling stations to monitor the Russian presidential election, 2012.

Summary

In this chapter we have learnt about web camera, video conferencing.

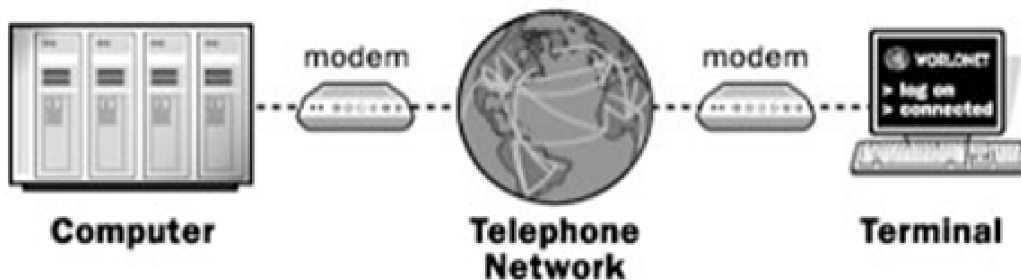


Key Words

- Video calling
- Video conferencing
- Magic Mirror

About Modems

With the evolution of technology, world has contracted in to a small village. One single click on your computer can connect you to any part of the globe. All of this has been possible due to a small electronic device called Modem. In this information guide we will discuss in detail about modem, its types, history, working and other important things. **What is a Modem?** Modem is an electronic device that converts computer's digital information into analog carrier signals and vice versa. Computers use modems to communicate with each other over a network. The word modem is derived from “**modulator-demodulator**” that defines the functions it performs. **How Modem Works?** Modems are computer hardware typically used to transmit digital data over a phone line. The working of modems can be easily understood from this. Modems always work in pair. The sending modem converts computer's digital information into specific frequencies compatible with the phone line, the process is called modulation. The receiving modem decodes the signal back into the digital information, the process is known as demodulation. In wireless modems, digital data is converted into radio signals and vice versa.



Origin Of Modems

The requirement of communication between distant computers led to the usage of phone line for data transmission. Since the phone lines were designed to carry analog information (voices) only and computers and its related network devices work in digital form, an interface was needed to bridge this gap, which could act as a converter between the two systems. The result came in the form of Modems.

Initially modems were primarily used to communicate data between terminals and a host computer. Later their use was extended to communicate between end computers. It

was around 1960 when modems came into existence. 1960s were the age of time-shared computers. A business would often buy computer time from a time-share facility and connect to it via a 300-bit-per-second (bps) modem. Now modems are used for different functions. Their applications include textual and voice mail systems, facsimiles, and they are integrated into cellular phones, pdas, and notebook computers enabling them to send data from anywhere. All these needs have increased the modemspeed up to 28.8 kbps.

Types of Modem

The classification of modems can be done on the basis of a number of criteria that includes line type, operation mode, synchronization, modulation, and range. Keeping aside all these types of modems, in this section we will discuss some of the most common and popularly used types of modems worldwide.

DSL (Digital Subscriber Line) Modem

DSL modems are the type of modems used to connect a PC to the Internet. These modems provide fast Internet access as they work on high-speed DSL connections that are considered much faster than dial-up Internet connections. The foremost advantage of DSL is its ability to use the phone line to make or receive calls while connected to the Internet. The traditional dial-up service can't provide this service without the use of a second phone line. DSL achieves this by adding a filter on the phone jacks in a location that will have telephones connected to them. A DSL modem provides similar service to that provided by a dial-up modem, the only exception is the high speed. But the setup and technological make up of DSL modems differs from the dial up. Other major difference is that DSL modems are external modems and connects to a computer via a USB or Ethernet port, while the dial-up modems are usually installed inside computer terminals.



ADSL (Asymmetric Digital Subscriber Line) Modem

ADSL modem provides faster downloading (getting data) than uploading (sending data). These modems have significant advantages over the dial-up modems. Like dial up modem it also uses a standard telephone line but it does not tie up the line. Thus the



telephone line can be used while accessing the Internet. ADSL service is like an “always on” connection that cannot be left connected indefinitely. ADSL modem is faster than a dial-up modem. Due to available bandwidth on the dedicated copper wire, ADSL modem can accommodate a telephone conversation. An ADSL modem needs ADSL service through an Internet Service Provider (ISP). It can’t work with a dial-up account.

SDSL (Symmetric Digital Subscriber Line) Modem

SDSL modems works on SDSL lines that provide equal bandwidth in both directions. Such types of modems are useful for those businesses which need to upload as well as download large files or programs to or from the Internet. These modems use voice traffic lanes to expand bandwidth, and therefore aborting the possibility of a conversation when the SDSL modem is connected. For this purpose a dedicated phone line is required.

Cable Modem

Cable modems are the types of modems that uses cable TV wiring instead of the phone line to provide Internet connectivity. They provides bi-directional data communication via radio frequency channels on cable television infrastructure. Cable modems mainly deliver broadband Internet access in the form of cable Internet taking the benefit of the high bandwidth of a cable television network. Cable modems work on the same principle as the DSL modem. One needs to have service from cable TV provider for Internet connectivity and also some network cables. The frequencies used for data traffic and TV traffic do not interfere each other. Cable modems are commonly deployed in Europe, North and South America, and Australia.



Sat Modem (Satellite Modem)

One of the least common types of cable modem is a satellite modem, or sat modem. It is a wireless modem converting digital data into radio waves to communicate with a satellite dish. This particular type of service is more expensive than other conventional types of Internet connectivity. Satellite modems can be of great help for businesses or people in rural areas that do not yet have DSL or cableservice offerings

Internal Modem

Internal modems are installed inside desktops or laptop computers, enabling the computer to communicate with other connected computers over a network. Internal modems are of two types, dial-up and WiFi. Dial up modem operates over a telephone line and needs a network access phone number and log-on credentials to do the connection. A wi-fi modem can connect wirelessly and without credentials in certain cases.

External Modem

All the modems initially used were external modems. They were in proprietary use for decades prior to 1981, which marks the release of the first affordable and practical modem for public use. External modems are useful in situations when no internal slots are available, or if the modem needs to be shared between computers that are not networked. An external modem connected to a desktop system can easily be disconnected and connected to different desktop or laptop. External modems for dial-up service are easy to setup and come with instructions. They are inexpensive and available at all places where computers are sold. These modems also include fax capability at the software level.

Summary

In this chapter we have understood about modem and types of modem.



Key Words

- Satellite
- Modulator
- Demodulator
- Asymmetric
- Subscriber

Learning Objectives

In this chapter you will be able to understand about ;

- The factors affecting the computer performance.

You may be wondering why your computer is slow at times and there are other times when it is fast in processing. This could be caused by a number of factors. They include: the speed of the CPU, the space on the hard disk, the size of the RAM, the type of the graphics card, the speed of the hard disk, if the computer is multitasking, the defragmenting files.

- 1) **The speed of the CPU** The speed of the CPU is also known as the clock speed of the CPU. The clock speed of the CPU is the frequency of which the processor executes instructions or the frequency by which data is processed by the CPU. It is measured in millions of cycles per second or megahertz (MHz). If the Clock speed of the CPU is fast then definitely the performance of the computer will be affected positively, in other words the computer will carry out processing functions at a faster pace.
- 2) **The size of the RAM (Random Access Memory)** The RAM is referred to as the active part of the computer. This is because the RAM has the capability of storing data that the computer is currently using, because of the fact that it is fast to retrieve data stored in the RAM. With the definition above, a large RAM size will mean a faster computer performance and a smaller RAM size will result to slower computer performance.
- 3) **The speed of the hard disk.** The hard disk speed is defined as the rate at which material and content can be read and written on it. The hard disk speed of different hard disks is not consistent because they vary by manufacturer, drive type and the use of the hard disk. It therefore means that the higher the speed of the hard disk the faster the performance of the computer and vice versa.
- 4) **Hard disk space** The bigger the space on the hard disk will result to faster performance of the computer. The smaller the space on the hard disk will result in

a slower performance of the computer. The hard disk is filled with data this will use most of the memory leaving less memory for the operations of the processor.

- 5) Multiple applications running on the computer Multi-tasking tends to slow down the performance of the computer because memory is used to support more than one applications compared to when one application has all the memory to itself. This means that the more applications that are running the slower the computer will perform. Likewise if less or one application is running the performance of the computer will be faster.
- 6) Type of graphic card When it comes to quality of pictures and animations graphic cards are the main factors. So if a machine processes many graphics and it has a weak graphic card it will perform slower. This means that the more powerful the graphic card is the faster the performance of the computer.
- 7) Defragmenting files Files that are broken or it takes long to read them will mean that the computer will have to defragment them first. This will slow down the performance of the computer.

Summary

In this chapter we have understood about the factor effecting computer performance.

Chapter - 6 Configuring Windows for Networks

Learning Objectives

In this chapter you will be able to understand about ;

- Configure a printer in Windows
- Share a printer in Windows
- Add a network printer to your system
- Create a new user account
- Assign a password for a user account
- Change the name of the computer
- Change the name of a Workgroup

Configuring a Printer

A printer attached to a computer needs to be configured before you use it. You can configure a printer in Windows either for a printer attached to your computer or for a printer shared across the network.

Printer Driver refers to the software that enables applications such as MS Word, Notepad, and so on, to communicate with the printer regardless of its model.

To configure a local printer in Windows:

1. Choose Start ? Printers and Faxes option. The Printers and Faxes window is displayed. The window will be empty if there are no printers attached to it.
2. Double-click Add Printer icon. The Add Printer Wizard is displayed.
3. Click Next to continue with the wizard.
4. Choose the Local printer option to configure a local printer or the option Network printer to configure a network printer and click Next.
5. Click Next to detect a printer manually.
6. Select a port to connect the printer and click Next.

7. In the dialog box displayed, select the model and the manufacturer of your printer. Based on the selected model and the manufacturer, the corresponding printer software is automatically installed.
8. Type a name for the printer in the Printer name text box and click Next.
9. Select Do not share this printer in Printer Sharing Dialog Box. Click Next.
10. Select the Yes option to print a test page, otherwise, click No.
11. Finally, click Finish to complete the configuration.

This will configure a local printer attached to the computer.

Sharing a Printer

Hardware resources, like printers are expensive. Connecting a printer to each computer on a network is very expensive. Hence, a single printer on a network can be shared and accessed by other users in a LAN. Printers can be shared after they are configured, by following the steps given below.

To share a printer:

1. Choose *Start ? Printers and Faxes option*.
2. Right-click the printer icon which you want to share.
3. Select the *Sharing* option from the pop-up menu.
4. Select the *Share this printer* option and specify a *Share name*.
5. Click the *Apply button* and then click the *OK button*.

User Accounts

A *user* is a person who uses the resources on a computer and the network. Each user has a *user account*. A user account contains information such as user name and password for the account. User name is an identification of a user on a network and should be unique.

Types of user accounts :

The *Computer Administrator* account is used to

- Create, change and delete accounts
- Make system-wide changes

- Install various programs and access files
- The *Limited* account is used to
- Change or remove your password
- Change your picture, theme and desktop setting
- View the files that you created
- View the contents of the Shared Documents folder

Users with limited account will not be able to install certain programs. Therefore, it is better to choose the computer administrator account type.

Logging in as Different User

There can be many user accounts for a single PC. Each user can have their own files and documents. The Log Off option is used to log off from the current user. Once you have logged off, you can log in as a new user, but when you log off from the current account, you have to close all applications running currently. Only after you exit all applications, you will be able to log off.

To log in as a different user:

1. Click *Start ? Shut Down...* Option • Select *Log Off User name*.
2. Click the OK button.

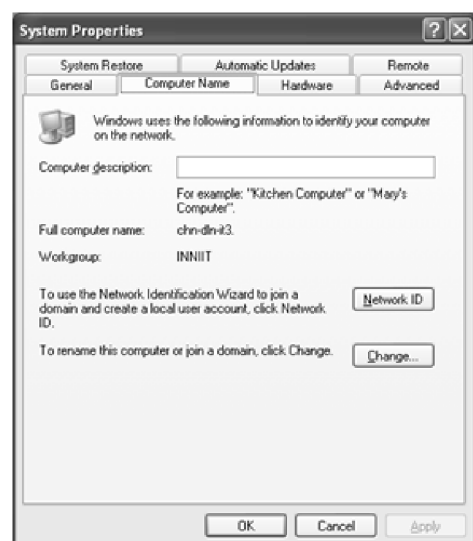
The Windows desktop will be opened for the new user.

Computer Name

Every computer on a network should have a unique name. This helps you identify the computer easily in the network. A computer name can consist of alphanumeric characters with no spaces in between. The name specified at the time of installation can be changed at any point of time.

To change the name of a computer:

1. Right click the *My Computer* icon on the desktop.



System Properties dialog box

- From the pop up menu displayed, select the Properties option.
- System Properties dialog box is displayed.
- Select the Computer Name tab from the System Properties dialog box.
- Click the Change... button.

A Computer Name Changes dialog box is displayed.

- Type a new name in Computer Name text box, and click the OK button.

A message asking you to restart the Computer to apply the changes is displayed.

- Click the *OK button to close the System Properties* dialog box. A message will be displayed where the system will ask if you want to restart the machine.
- Click Yes and restart the computer since the change will take effect only when you restart the computer.



The Computer Name Changes dialog box

Workgroup Name

A *workgroup* is a logical grouping of computers, which enable users to browse the network.

Consider a network of 20 computers. Let the name of the workgroup be 'Workgroup1'. Here, assume that you want to isolate the last five computers and form a separate workgroup. This can be easily implemented by changing the workgroup name for the last five computers. They will now form a new workgroup and will not be a part of 'Workgroup1'.

To change the workgroup name:

- Right click the *My Computer* icon on the desktop.
- From the pop up menu displayed, select the Properties option.
- System Properties dialog box is displayed.
- Select the Computer Name tab from the System Properties dialog box.

5. Click the Change....button.

A Computer Name Changes dialog box is displayed.

6. Type a new name in the Workgroup text box of Member of section, and click the OK button.

A message asking you to restart the Computer to apply the changes is displayed.

7. Click the *OK button to close the System Properties* dialog box . A message will be displayed where the system will ask if you want to restart the machine.

8. Click Yes and restart the computer since the change will take effect only when you restart the computer.

Summary

- A printer can be configured either for a printer attached to a local computer or for a printer shared across a network.
- A printer can be configured using the Add Printer Wizard.
- A printer on a network can be shared and accessed by users on a network.
- To access a printer attached to another computer on the network, the printer should be added to your local computer.
- A user is a person who uses the resources on a computer and network. Each user has a user account. The user account contains information such as user name and password for the account.
- The two types of user accounts that can be created are Computer Administrator and Limited.
- Creating a password for a user account restricts the access to the account, only to users who know the password.
- Every computer on a network should have a unique name. This helps you identify the computer easily on the network. This name can be changed using the System option.
- A workgroup is a logical grouping of computers, which enable users to browse the network. Computers can be grouped into various workgroups and the name of the workgroup can be changed.



Teacher Activity

Demonstrate printer configuration.



Test Your Knowledge

Fill in the blanks

1. Refers to the software that enables applications such as MS Word to communicate with the printer.
2. A network printer can be added to your computer by using the wizard.
3. A user account contains information such as and for the account.
4. The account is used to install various programs.
5. The limited account is used to view the contents of the folder.
6. A password can have _____ , _____ and _____ .
7. The option is used to switch between users without closing the currently running programs.
8. A computer name can consist of characters.
9. A is a logical grouping of computers, which enable users to browse the network.
10. The user account is used to change or remove your password.

Learning Objectives

In this chapter you will be able to understand about ;

- Different types of virus.

Malware

Malware, or malicious software, is a term that encompasses any software designed to corrupt one's computer. Viruses, trojans, worms, and spyware are some examples of malware. The entire purpose of malware is to disrupt or deny the computer's ability to function properly, and possibly access private information such as credit card numbers, passwords, social security numbers, or tax information that one may have saved in a folder. In the following sections we will discuss some of the common types of malware.

Computer Viruses

A computer virus works just like a virus would in the human body. It is an infected file that makes its way onto your computer. It then reproduces itself, spreading throughout your computer and even onto other computers by attaching itself to files shared on a network and accessed by other people.

To initiate the ability to reproduce, a virus will hide in executable files, in other words, in files that run programs. The virus may lie dormant in this file until you actually run the program, at which point the virus will also start running, reproducing, and wreaking havoc on your computer. An example scenario would be that you download a game or computer program from a pirating website or program, such as Limewire. Once downloaded, you click to run the executable file, unknowingly downloading the virus that is attached to your computer.

Trojans

You may remember the story of the Trojan Horse in mythology. This was a large wooden structure built and given to Troy by the Greeks as a "peace" offering. The Trojans accepted the gift, and the horse was wheeled into the walled city. Hidden inside were Greek soldiers, who attacked the sleeping Trojans from inside their own walls. A trojan horse, or trojan, is just that, a seemingly innocuous file containing hidden malicious code waiting to

attack your computer when you least expect it. This is not a virus, as it does not reproduce itself, but it is still just a dangerous. A trojan file will appear to be a simple computer program that you have downloaded, for example a program that states it will remove malware, when it too is malware. Once running, it will appear to be removing the malicious software while at the same time trying to access and steal your personal information.

How to Avoid Downloading Malicious Software

The most obvious thing to do to avoid downloading malware onto your computer is to not download at all. However, this is not always possible since sometimes you need to download software required to do your work such as Open Office, Java, Adobe, etc. So, when downloading a file, always make certain you are accessing the file from a reputable site. CNET.com is one site example that is normally quite secure to download programs from. Be particularly careful to avoid downloading anything from advertisements and pop-ups.

Also, make certain that your computer is running an up-to-date scanning software. The purpose of these programs are to help catch, contain, and sometimes removal malicious software that may have made it on to your computer. These programs may be free, such as AVG free, or have paid versions that require a yearly subscription, such as Webroot.



Test Your Knowledge

Answer the following.

- 1) List at least two ways, not mentioned in the lesson, a virus may get on to your computer. What steps could have been done to avoid this happening?
- 2) What does it mean when a virus scanner has “quarantined” an infected file on your computer?
- 3) What does it mean to reformat your computer? Why would someone have to do this?

Learning Objectives

In this chapter you will be able to understand about ;

- Rename files and folders
- Copy and move files and folders
- Select multiple files and folders
- Delete files and folders
- Empty the Recycle Bin
- Restore a file from the Recycle Bin

Introduction

Windows allows you to change file names. Windows also allows you to copy, move and delete files and folders. Such actions performed on files are called file operations. These actions can be performed on folders also. All these operations can be performed by using either *My Computer* or *Windows Explorer*.

Renaming Files and Folders

Changing the names of files or folders is called renaming.

In the previous class, you had created a text file having your friends' names and addresses. The name of the file is "New Text Document". Suppose you want the file name to be changed to "Addresses".

To rename a file:

1. Select the "New Text Document" file.
2. Click the *File* menu.
3. Click the *Rename* option (refer Figure 1).

The file name "New Text Document" is highlighted (refer Figure 2).

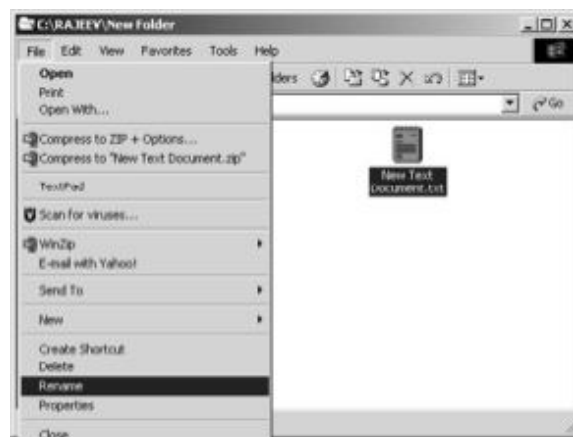


Figure 1: The Rename option on the File menu

4. Type the name “Addresses”.

5. Press the *Enter* key.

The file name changes to “Addresses”.

Alternatively,

1. Right-click the file.

A pop-up menu appears.

2. Click the *Rename* option as shown in Figure 3.

Tip : Alternatively, press the *F2* key to rename a file.

Suppose you now want to change the name of your folder to “MyFolder”. Renaming a folder is similar to renaming a file. You can use any of the above methods to rename the folder.

Copying a File or Folder

You now know how to create a folder and then, create a file in it. Suppose you want to make a copy of the text file “Addresses” on the desktop.

You can copy files and folder using *My Computer* or *Windows Explorer*. To copy the file by using *Windows Explorer*:

1. Open the *Windows Explorer* window.

2. Select the file “Addresses”.

3. Click the *Edit* menu.

4. Click the *Copy* option (refer Figure 4).

5. Select the *Desktop* icon in the left pane of the window.

6. Click the *Edit* menu.

7. Click the *Paste* option.

Your file has now been copied to the desktop.

Note : Alternatively, you can right-click the file or folder and select the *Copy* and *Paste* options on the pop-up menu.



Figure 2: Highlighted file name

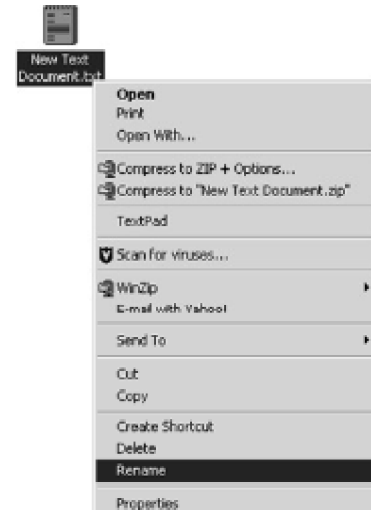


Figure 3: The Shortcut menu



Figure 4: The Copy option on the Edit menu

Moving Files and Folders

Your folder is currently in the C: drive. Suppose you want it to be moved to the *My Documents* folder.

To move files or folders by using *Windows Explorer*:

1. Select the “MyFolder” folder.
2. Click the *Edit* Menu.
3. Click the *Cut* option.
4. Select the *My Documents* folder.
5. Click the *Edit* menu.
6. Click the *Paste* option.

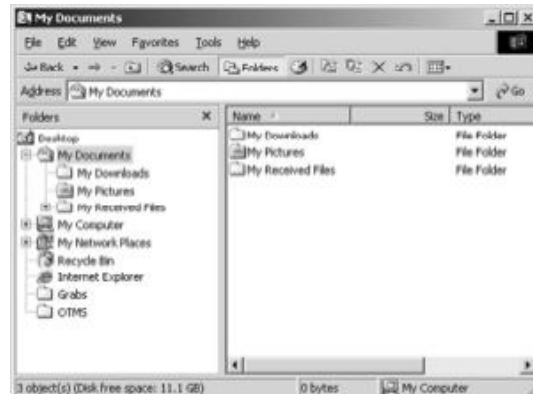


Figure 5 : Folder after being moved into My Documents

Now, the *MyFolder* folder has been moved to the *My Documents* folder (refer Figure 5).

Note : Alternatively, right-click the file or folder and select the *Cut* option on the pop-up menu.

Selecting Multiple Folders and Files

Consider the following situation.

You have created a new folder called “My Pictures” under the My Documents folder. You also created a number of drawings in Paint and saved the pictures under this folder.

You want to make a copy of all these pictures on the desktop. But it is very time-consuming to copy the files one by one. Windows helps you by allowing you to copy or move multiple files at the same time. For this, you must know how to select multiple files.

To select all files and folders within a folder:

1. Open the folder. The contents of the folder appear in the right pane.
2. Click the *Edit* menu.
3. Click the *Select All* option (refer Figure 6).

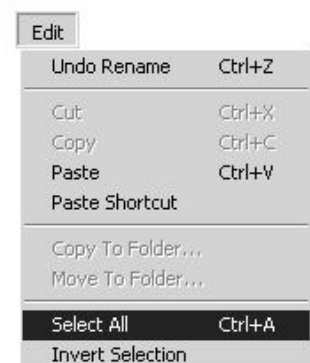


Figure 6: The Select All option

All the files within the “My Pictures” folder are selected as shown in the Figure 7.

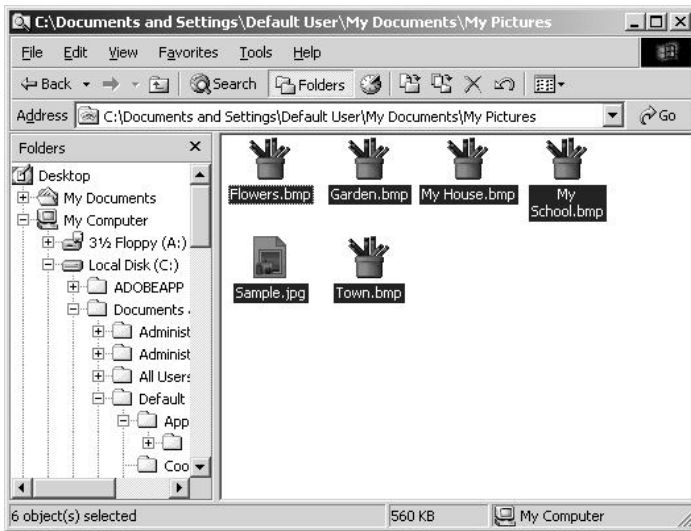


Figure 7: Files selected using the Select All option

Now, you have to use the *Copy* and *Paste* options only once to copy the pictures to the desktop.

Tip : Alternatively, press the *Ctrl* and *A* keys together to select all the files and folder in a folder.

Consider a situation where you do not want to select all the files but want to select the last three files. The *Shift* key can be used to select consecutive files and folders.

To select consecutive files by using the *Shift* key:

1. Click the first item to be selected, in this case, the file “My House”.
2. Press and hold the *Shift* key.
3. Click the last item to be selected, in this case, the file “Town”.

All the files between the first and the last item you clicked are selected. The selected files and folder are shown as in Figure 8.

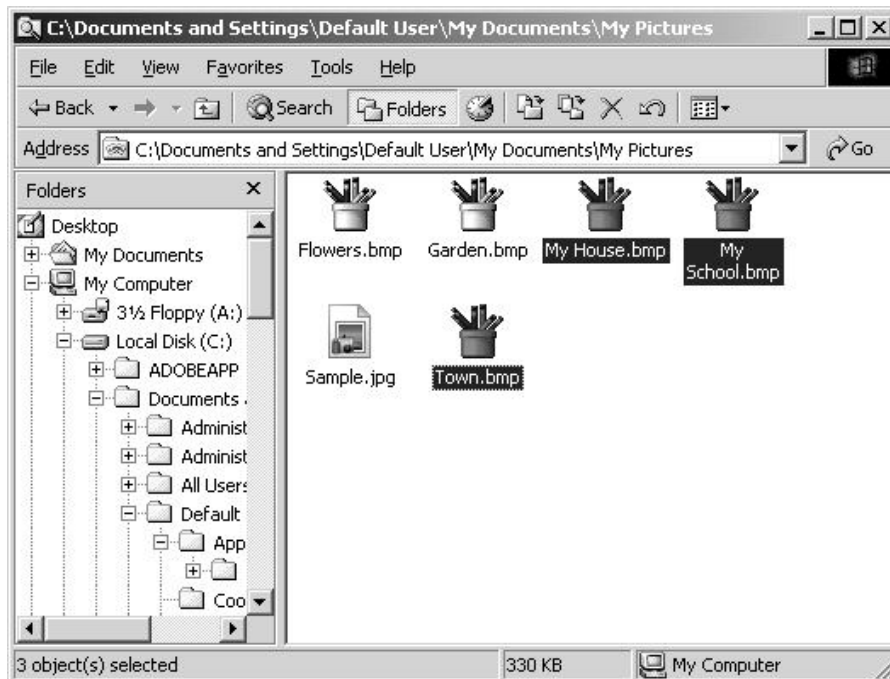


Figure 8: Files selected using the Shift key

Windows also provides an option to select all the files and folder within a particular folder.

You just saw how to select consecutive files from a folder. Windows also allows you to select non-consecutive files in a folder. This is possible by using the *Ctrl* key.

To select multiple files by using the *Ctrl* key:

1. Click the first item to be selected.
2. Press and hold the *Ctrl* key.
3. Click each of the items to be selected.

The selected files and folder are shown as in Figure 9.

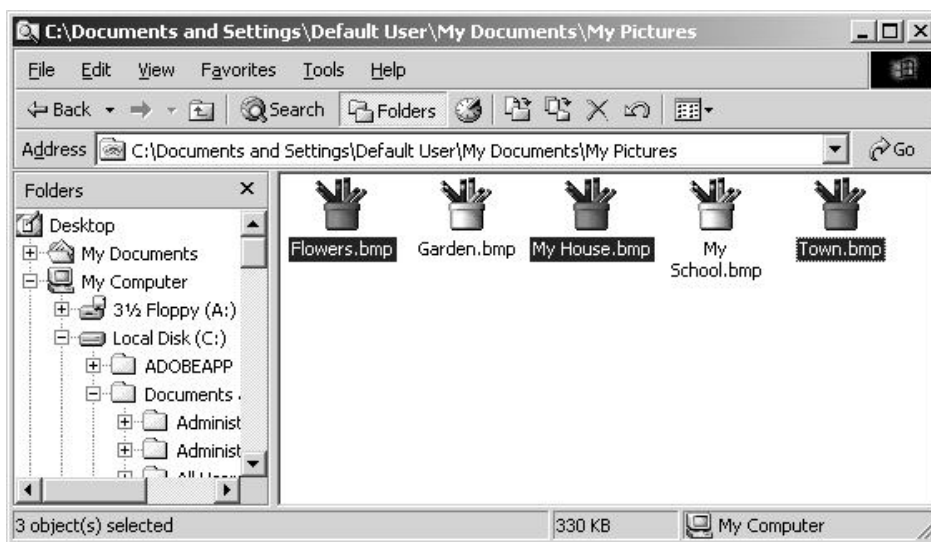


Figure 9: Files selected using the Ctrl key

Deleting Files and Folders

Windows allows you to delete files and folders on your computer. Deleting files that you no longer need frees disk space and also keeps your data well organised.

For example, if you want to delete the file “Addresses” from the folder “MyFolder”:

1. Select the file you want to delete.
2. Click the *File* menu.
3. Click the *Delete* option.

A dialog box appears as shown in Figure 10 where it asks for confirmation before deleting the file.



Figure10: Confirm Folder Delete dialog box

The Recycle Bin

The Recycle Bin holds all the deleted items. When a file or folder is deleted, it is deleted from the folder or drives and stored in the Recycle Bin.



Figure 11: The Recycle Bin icon

1. Double-click the *Recycle Bin* icon (refer Figure 11).
2. The Recycle Bin window is displayed (refer Figure 12).

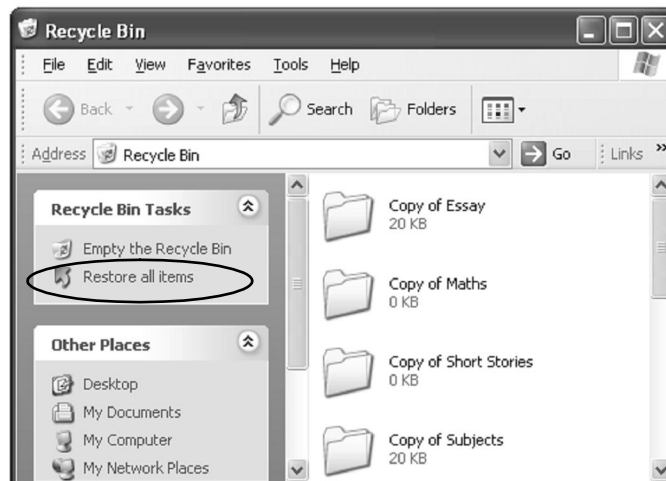


Figure 12: The Recycle Bin window

To restore a file or folder back to its original location,

1. Select the file or folder.
2. Click the *File* menu.
3. Click the *Restore* option.

Alternatively, right-click the file or folder and click the *Restore* option on the shortcut menu.

Tip : You can also restore a file or folder by selecting it and clicking the *Restore this item* option from the left pane.

To restore all the items,

Click the *Restore all items* option from the left pane (refer Figure 12).

To delete a file or folder permanently from the Recycle Bin, perform the following steps,

1. Select the file or folder.
2. Click the *File* menu.
3. Click the *Delete* option.

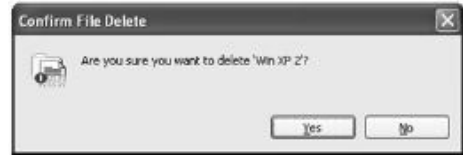


Figure 13: Warning dialog box

Alternatively, select the file and press the *Delete* key.

A warning dialog box is displayed asking for confirmation before deleting a file or folder. If you really want to delete the file or folder, click *Yes*. Otherwise, click *No*.

Note : Once you delete the file from the Recycle Bin, you cannot restore it again.

To empty the Recycle Bin,

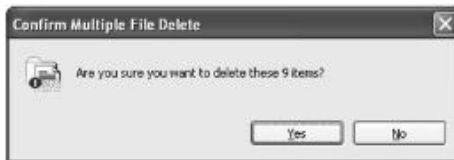


Figure 14 : Warning dialog box

1. Click the *Empty the Recycle Bin* option on the left pane.

A warning dialog box is displayed asking for the confirmation. (refer Figure: 14)

If you want to empty the Recycle Bin, click *Yes*. Otherwise, click *No*.

Sharing a Folder

A folder is a means of organising programs and documents on a disk and can hold both files and additional folders within it. Folders can contain different types of files such as documents, music, pictures and programs. Windows XP allows you to share a folder on the network. Sharing allows other users to access the folder on your computer through the network.

To share a folder,

1. Click the Start button.
2. Click the My Computer option.
3. Double-click the drive where the folder is placed.
4. Select and right-click the folder that has to be shared.
5. Click the *Sharing and Security* option on the shortcut menu.

Select this option to share the folder on the network



Figure 15 : Folderpropertiesdialogbox

The *Properties* dialog box of the folder is displayed (refer Figure 15).

6. Select the *Share this folder on network* checkbox.

If you do not want the network users to make any changes to this folder, clear the *Allow network users to change my file* checkbox.

7. Click the *Apply* button, and then click the *OK* button.

Note : The *Sharing* option is not available for the Documents and Settings, Program Files and Windows system folders.

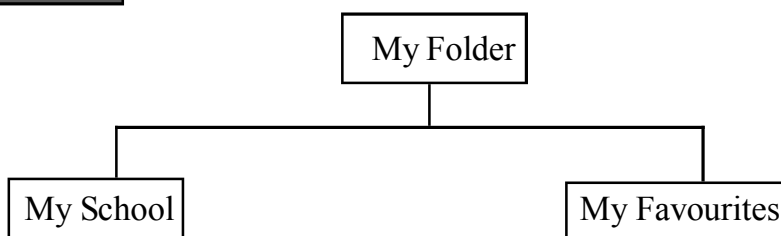
You can also change the shared name of the folder. To change the name of the folder on the network, enter a new name in the *Share Name* text box. This will not change the name of the folder on your computer, but will only the name as others will see.

Now You Know :

You can rename, copy, move or delete a folder using Windows Explorer.



Demonstrate all menu options.



1. Create a folder in the C.
2. Rename the folder as “MyFolder”.
3. Create two folders under “MyFolder” namely, “MySchool” and “MyFavourites”.
4. Create two Text documents under the folder “MySchool” and rename it as “Marklist” and “Friendslist” respectively.
5. Create a Paint file under the “MyFavourites” folder and rename it as “FavouritePicture”.
6. Move the “FavouritePicture” file from the “MyFavourite” folder to the “MySchool” folder.

7. Copy the “Friendslist” file from the “MySchool” folder to the “MyFavourite” folder.
8. Make a copy of the “Friendslist” file in the “MyFavourites” folder.
9. Delete the “Friendslist” document in the “MySchool” folder.
10. Open the Recycle Bin and restore the deleted file.
11. Permanently delete the copy of the “Friendslist” file in the “MyFavourites” folder.
12. Share the folder “My Folder”.



Test Your Knowledge

Fill in the blanks

1. You can change the name of your folder using the _____ option.
2. The _____ option is used to copy a file from one place to another.
3. To select the files successively the _____ key is used.
4. When a file is deleted from a folder, they are moved to the _____.
5. The _____ option is used to delete all the files and folders in the Recycle Bin.
6. Data, information and other resources can be shared across machines by sharing the _____.

True or False

1. You cannot delete a folder, if it is going to hold few files.
2. You can change the name of a folder or file.
3. The *Ctrl* key is used to select the files and folders that are consecutive.
4. The *Select All* option is used to select all the files and folders.

Learning Objectives

In this chapter you will be able to understand about ;

- Add custom animations to slide objects
- Time your presentation

Adding Custom Animation

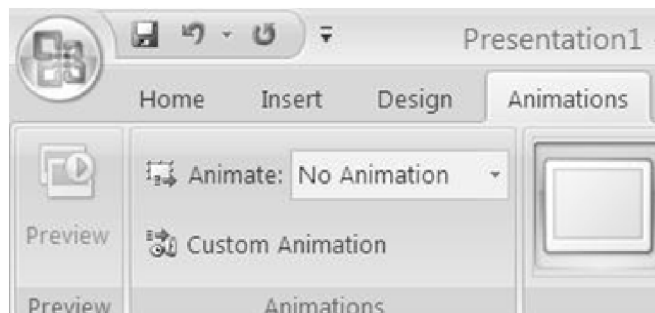
Rahul shows the presentation created by him to his father. He has also added slide transitions to his slides. His father suggests that he add some more special effects to the presentation. Rahul asks his father how he could do this. His father tells him that he can animate the different objects present in the slides.

The *Custom Animation* option is used to create your own animation schemes. This allows you to control how and when you want an item to appear on a slide, during your presentation. For example, you can make the slide text fly in from the left when you click the mouse.

Custom animations make the presentation more attractive and professional since the user is allowed to choose suitable effects for each item on the slide.

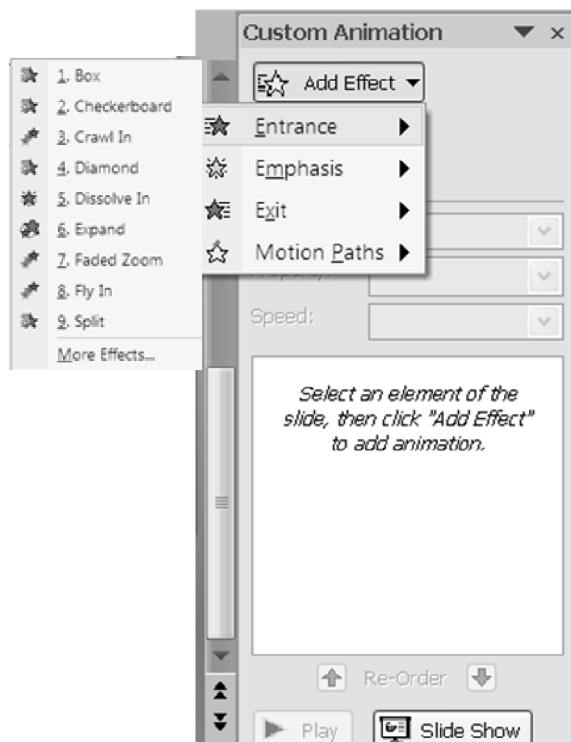
To apply custom animation:

1. Select the slide that has the text or objects you want to animate.
2. Select the object you want to animate.
3. Click the Animations tab.
4. Click the *Custom Animation* button from the Animations group.



The CustomAnimation button in the Animations group

The *Custom Animation* task pane appears.



The Custom Animation task pane

- *Exit*: Used to set the exit animation for an object.
- *Motion Paths*: Used to move the objects in a specific path.

Each of the above options displays some commonly used effects. You can view the other effects by clicking the *More Effects* option available under each category.


6. Select the required *Property* option.

The *Property* drop-down list is used to specify the direction from which the animation should play.

7. Select the required *Start* option.

The *Start* drop-down list is used to specify the timing of the animation event in relation to the other events on the slide. The choices are as follows.

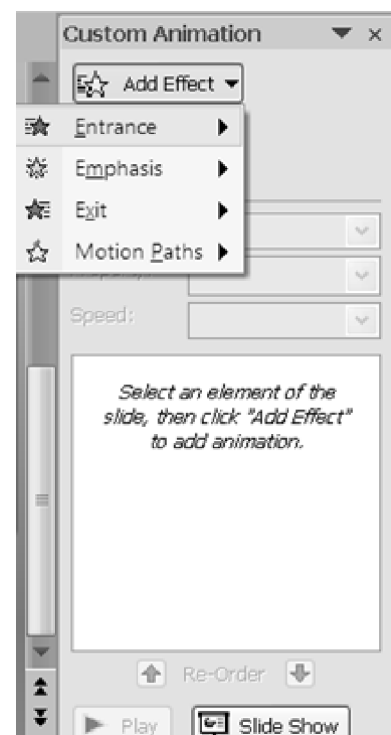
- On Click – Animation event begins when you click the mouse on the slide.

5. In the *Custom Animation* task pane, click the  **Add Effect** button and choose one or more animation effect(s) for the objects.

The *Add Effect* list allows you to add one or more custom animation effects to the selected object.

The options available in the *Add Effect* list are:

- *Entrance*: Used to set the entry animation for an object.
- *Emphasis*: Used to set the animation for an object, which starts after the object enters the slide.




The Add Effects list

- With Previous – Animation sequence begins at the same time as the previous item in the list (that is, one click can execute two animation effects).
 - After Previous – Animation sequence begins immediately after the previous animation in the list is finished playing, that is, no additional click is required to make the next sequence start.
8. Set the speed in the *Speed* list.

The *Speed* option is used to specify the speed level at which the text or object should appear during the show. The various speed levels are *Very Slow*, *Slow*, *Medium*, *Fast* and *Very Fast*.

9. Change the sequence of animations if you want to, by using the *Re-order* option. The objects along with the effects appear in the Custom Animation list, in the order you apply them. To change the order, select the item you want to move from the list and drag it to another position in the list.
10. Click the *Play* button to preview the animation.

Note : If you want to remove an effect, select the animation item in the Custom Animation list and then click the  Remove button in the Custom Animation task pane.

Motion Paths

Motion Path refers to a path that a specified object or text will follow as part of an animation sequence for a slide.

To apply or draw a motion path:

1. Select the text or object you want to animate.
2. In the *Custom Animation* task pane, click the *Add Effect* button and select the *Motion Paths* option.
3. Apply a preset motion path or create a custom motion path by using the options available under it.

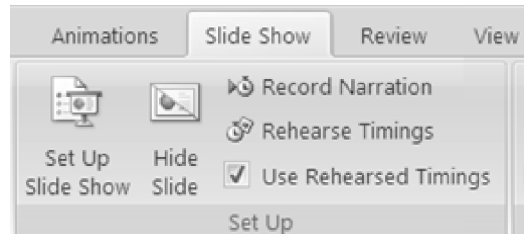
When you view the slide show, the animation follows the sequence specified in the motion path.

Timing Your Presentation

Rahul asks his father what he must do if he wants the presentation to move forward on its own. His father suggests that he use the rehearsal mode. The rehearsal mode lets you adjust the timing so that viewers have enough time for each slide.

To apply timings for slide transition:

1. Click the Slide Show tab.
2. Click the *Rehearse Timings* button from the *Set Up* group.



The presentation enters the Slide Show mode and the *Rehearsal* dialog box appears.

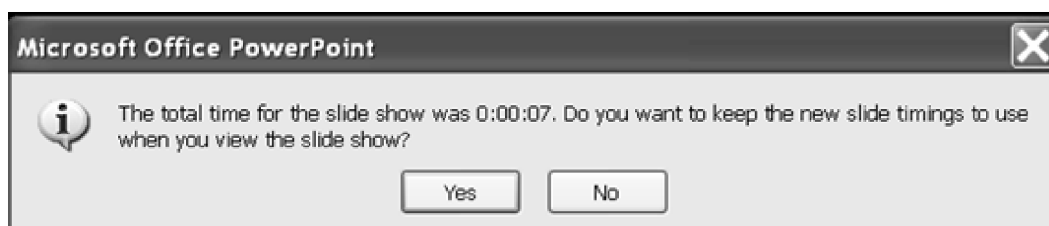
The Rehearse Timings button from the Slide Show tab



The Rehearsal toolbar

3. Type the amount of time for which the current effect should appear.
4. Click the arrow button on the *Rehearsal* dialog box to move to the next effect.
5. Repeat steps 3 and 4 until all the effects have been timed.

After the timings for all the slides in the presentation have been rehearsed, a dialog box as in appears. It displays the total time for the slide show and the question “Do you want to record the new slide timings and use them when you view the slide show?” is displayed. Click the *Yes* button to record the new timings or the *No* button to retain the previous timings.



Microsoft PowerPoint dialog box

Summary

- *Custom Animation* can be applied to text and objects on a slide.
- The rehearsal mode lets you adjust the timing so that viewers have enough time for each slide.



Teacher Activity

Demonstrate animation effects.



Student Activity

1. Create a presentation on „Pollution” :
 - a) In the first slide, insert the title „ ‘Pollution’”.
 - b) In the second slide, list the various „types of pollution” .
 - c) In the third slide, list the Causes for each type of pollution.
 - d) Open the Custom Animation task pane.
 - e) Apply the, Diamond Out custom animation effect to all the objects in the presentation.
 - f) Set the Speed to ‘Slow’ , and the *Sound* to „ ‘Wind’”.
 - g) Set the *Advance Slide* option to „ ‘Automatically’” after and set the timing to 3 seconds.
 - h) Time your presentation.



Test Your Knowledge

Fill in the Blanks

1. To add animation effects, the _____ option is used.
2. To move an object in a specific path, the _____ option is used.
3. The _____ option is used to change the sequence of animation.
4. The _____ mode lets you adjust the timing of each slide.
5. The _____ drop-down list provides the options for specifying the direction of animation.

Chapter - 10 Creating Tables Word Document

Learning Objectives

In this chapter you will be able to understand about ;

- Create a table
- Enter data into a table
- Add rows at the end of a table
- Insert rows and columns
- Change the row height and column width
- Repeat the table heading in subsequent pages
- Delete rows and columns
- Select and delete a table

Creating a Table

Your teacher wants to keep a record of the test results in a Word table since a table helps to organise and present data. It would also be more convenient to perform calculations and sort the data present in a Word table than in a register. Since she is not familiar with Word, she gives you the task of creating a table in Word to maintain the test results.

Note : A table is made up of rows and columns of cells, in which you can enter data or insert graphics.

To insert a table in a Word document:

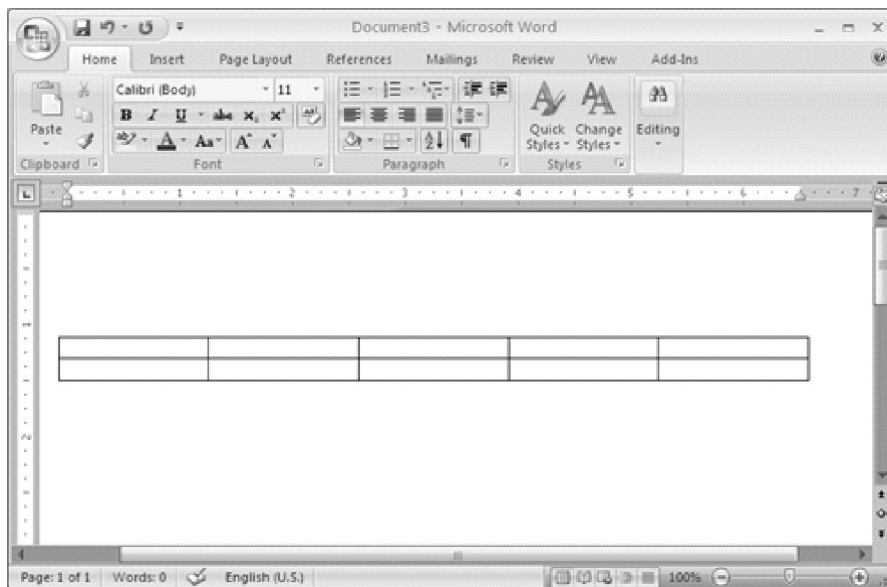
1. Place the cursor at the position where you want the table.
2. Click the *Insert Tab*.
3. Select the *Table list from the Tables group*.
4. Select the Insert Table option from the submenu.

A Insert Table dialog box is displayed.

5. Click the *Table* option. The *Insert Table* dialog



box appears. You can specify the number of rows and columns in the *Table size* section. Word inserts a table with 2 rows and 5 columns at the cursor position.



The inserted table

Entering Details into the Table

The table has been created. You have to enter the name and marks scored in each subject for all the students into the table. The first row will have the column headings and the second row onwards will have the details about each student.

To enter values in the table:

1. Place the cursor inside the first cell of the first row.
2. Type the heading "Name".

Name				

Adding text in a table

3. Press the *Tab* key.
4. The cursor moves to the next cell, that is, the second cell of the first row.
5. Type the heading "English".
6. Press the *Tab* key again. The cursor moves to the next cell.

7. Similarly, enter the subject names Telugu, Maths and Science to the third, fourth and fifth cell of the first row respectively.
8. After entering the subject name in the last column of the row, press the *Tab* key to move the cursor to the first cell of the next row.
9. Type the first student name (Ajay) in the first cell of the second row.
10. Press the *Tab* key to move the cursor to the next cell in the same row.
11. Type the marks scored in English in the cell.
12. Similarly, enter the marks scored by the student in the other subjects in the second row of the table.

Name	English	Telugu	Maths	Science
Ajay	80	80	97	75

The table with the marks of one student

Adding Rows at the End of a Table

The table has the marks of one student. To enter the name and marks of the other students, you have to add more rows to the table.

To add rows to a table:

1. Place the cursor in the last cell of the last row, in this the second row.
2. Press the *Tab* key.

A new row gets added to the table as shown.

Name	English	Telugu	Maths	Science
Ajay	80	80	97	75

Table after adding a row

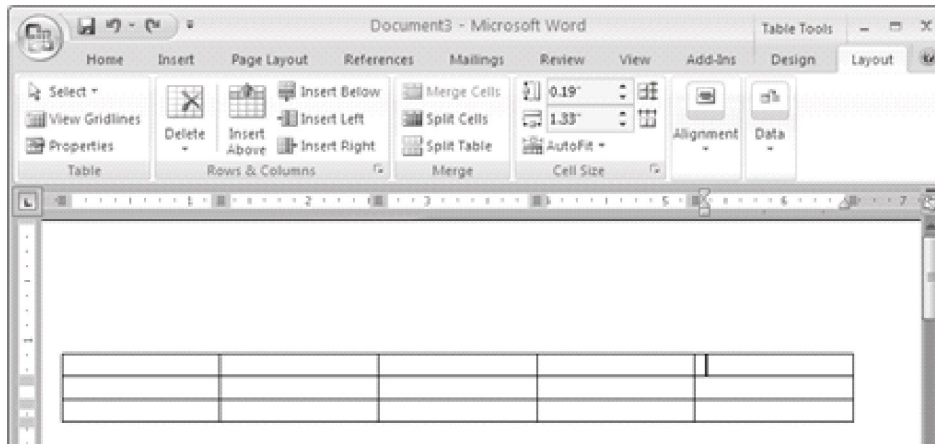
The marks of the second student can now be entered in the new row. Similarly, you can add more rows to the table and enter the name and marks of all the other students.

Inserting Rows and Columns

You want to enter the marks scored in “Computer Science” also to the table. The column for this subject has to appear before the last column, “Science”. You can do this by inserting a column to the table.

To insert a column:

1. Place the cursor in the column titled “Science”.
2. Click the Layout tab.
3. Choose the *Insert Left* option from the Rows & Columns group.



Insert Left option in the Layout tab

A new column gets inserted Type the heading “Computer Science” in the first cell of the column and enter the Computer Science marks for each student in the corresponding row.

Name	English	Telugu	Maths	Computer Science	Science
Ajay	80	80	97		75
Ameeta	85	82	92		85
Arun	76	72	81		85

Table with the inserted column

You realise that you missed out the mark details of a student, Sathya. Since you are entering names in alphabetical order, you have to insert a row before the last row of the table to add the mark details of that student.

To insert a row:

1. Place the cursor in the row before which you want to insert a row, in this case, the last row.
2. Click the Layout tab.
3. Choose the *Insert Above* option from the Rows & Columns group.

A new row gets inserted before the last row. Now, add the name and marks of the student to the table.

Meena	80	83	90		78
Naresh	73	70	65		60
Sujatha	76	87	79		67

The inserted row


Note : You can also place the cursor in the row after which you want to insert a row. In such a case, you have to select the *Rows Below* option.

Changing the Row Height and Column Width

In case the contents of a column do not fit within the column, the column width can be increased. You will notice that the heading “Computer Science” does not fit within the column. So, the column width has to be increased.

To resize a column:

1. Move the mouse pointer over the border of the column, which you want to resize.

The mouse pointer changes to a double-headed arrow .

2. Click and drag the boundary until the column width changes to the width you want.


By increasing the column width of one column, the column width of the immediate column will decrease. So, you can change the column width of the other columns in the same way.

Name	English	Telugu	Maths	Computer Science	Science
Ajay	80	80	97		75
Ameeta	85	82	92		85
Arun	76	72	81		85

The table after resizing the columns

Similarly, you can resize rows too. To resize a row

1. Move the mouse pointer over the border of the row, which you want to resize.

The mouse pointer changes to a double-headed arrow .

2. Click and drag the boundary until the row height changes to the height you want.

Similarly, you can change the row height of the other rows. In the following figure, all rows except the first row have been resized .

Name	English	Telugu	Maths	Computer Science	Science
Ajay	80	80	97		75
Ameeta	85	82	92		85
Arun	76	72	81		85

Table after resizing rows and columns

Repeating a Table Heading

You find that the table is so long that it continues to the next page. You want the heading to appear in the second page also.

To repeat the table heading in subsequent pages:

1. Place the cursor in any one of the cells of the heading column.
2. Right click.
3. Select the Table properties option.
4. From the Row tab, check Repeat as header row at the top of each page.

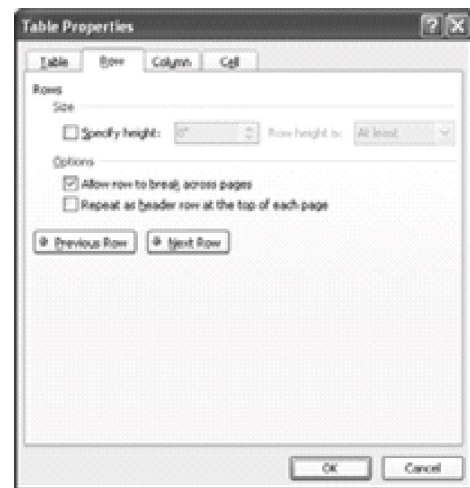


Table with the heading repeated

Word repeats the table heading on the second page.

Even if the table continues to the third page, the heading will appear in that page.

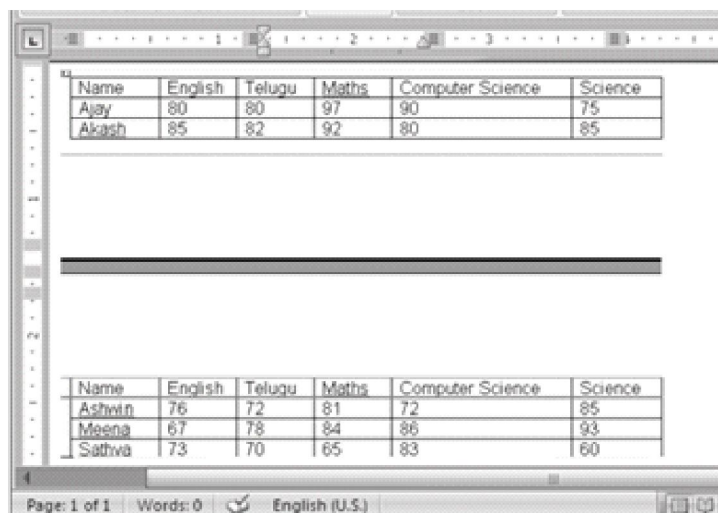


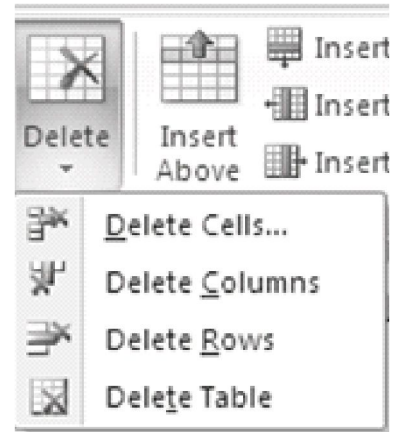
Table with the heading repeated

Deleting Rows and Columns

You entered the mark details of the student “Suresh” twice by mistake. The repeated row has to be removed from the table.

To delete a row :

1. Place the cursor in the row, which you want to delete.
2. Click the Layout tab.
3. Click Delete list from the Rows & Columns group.
4. Select Delete Rows option.



The row gets deleted from the table. Similarly, you can delete columns. To delete a column:

1. Place the cursor in the column, which you want to delete.
2. Click the Layout tab.
3. Click Delete list from the Rows & Columns group.
4. Select Delete Columns option.


The column gets deleted from the table.

Deleting a Table

You can not only remove rows and columns from a table but also remove an entire table and its contents from a document.

To delete a table:

1. Place the cursor in the table, which you want to delete.
2. Click the Layout tab.
3. Click Delete list from the Rows & Columns group.
4. Select Delete Table option. The table gets deleted.

Note : Alternatively, to select the table, click the table move handle  that appears on the top left corner of the table.

Summary

- To insert a table, click the *Inset Tab* and select the Table list.
- The *Tab* key can be used to navigate between the cells of the table.
- To insert a row, click the *Rows Above* or *Rows Below* option from the Rows & Columns group
- To insert a column, click the *Columns to the Left* or *Columns to the Right* option from the Rows & Column group.
- The column can be resized using the double-headed arrow mouse pointer.
- The row can be resized using the double-headed arrow mouse pointer.
- The table heading can be made to repeat in the subsequent pages using the Repeat as header row at the top of each page option from the Table properties dialog box.
- To delete a row, click the Delete Rows option *from the Rows and Column group*.
- To delete a column, click the Delete Columns option *from the Rows and Column group*.
- To remove a table, select the table, select the Delete list from the Rows & Columns group and select Delete Table option from the Layout tab.



Teacher Activity

- Demonstrate all the options taught in this chapter to the students with examples.



Student Activity

1. Create a new document and type the following table.

Elements	Compounds
Are pure substances	Are pure substances made of one or more
Have their own properties	Have properties different from that of the constituent elements
Example: Sodium	Example: Sodium Chloride (Common Salt)

2. Insert a column for “mixtures”, a new row and enter the text as shown below.

Elements	Mixtur	Compounds
Are pure substances	Are combination of two or more substances	Are pure substances made of one or more substances
Cannot be divided into two or more substances	Divisible into constituent elements	Divisible into constituent elements
Have their own properties	Constituents retain their properties	Have properties different from that of the constituents elements
Ex: Sodium	Ex: Air	Ex: Sodium Chloride

3. Resize the columns and rows to make the contents clear.

4. Make the table headings repeat in the subsequent page.

5. Remove the last row and the last column.

6. Remove the table from the document.

 **Test Your Knowledge**

Fill in the blanks

1. While creating a table, the number of rows and columns are specified in the _____ dialog box.
2. The _____ key is used to navigate from one cell to another in a table.
3. The double-headed arrow mouse pointer is used to resize _____ .
4. To add rows or columns the _____ group of the Layout tab is used.
5. Clicking the table move handle will _____ the table.

Working with Charts**Learning Objectives**

In this chapter you will be able to understand about ;

- Create charts
- Modify charts
- Print charts

Introduction to Charts

Your teacher has asked you to create a chart that shows the class average in English, Maths and Science in the Marks.xlsx workbook.

A chart is a picture that is used to represent the data in a worksheet. As you know, a picture is worth a thousand words. A chart helps you to understand the data stored in a worksheet in a better manner.

Excel charts are simple to create. You can create an embedded chart or an independent chart sheet. An embedded chart is a chart that is a part of the worksheet that contains the data. A chart sheet is a separate sheet like a worksheet within a workbook.

The worksheet data and the chart that you create are linked. Due to this, whenever you change the data in the worksheet, the chart also changes automatically.

Excel lets you create various types of charts: Column, Line, Pie, Bar and Area. A simple chart has two lines

The horizontal line is called the X-axis and the vertical line is called the Y-axis. When two sets of information have to be compared, the data will be plot pertaining to one set in the X- axis and the other in the Y-axis in case of simple charts.

Creating Charts

To create a chart that shows the average mark in English, Maths and Science in the Marks.xlsx workbook, you must first find the class average for English, Maths and Science in cells B19, C19 and D19 respectively. Format the cells to hold only two decimal places.

Student	English	Maths	Science	Total	Average
Siva	87	90	80	257	85.667
Giri	80	80	80	240	80
Venkat	70	70	70	210	70
Nozeer	80	75	85	240	80
Ameer	80	85	70	235	78.333
Highest	87	90	85		
Lowest	70	70	70		
Average	78.5	80	77.5		

Selecting the data for preparing chart

To insert a Bar chart:

1. Select the cells containing the data for the chart.
2. Click the Insert tab.
3. On the Charts group the various types of chart icons are displayed along with their names.
4. Click on the Bar Chart icon.
5. All the subtypes of Bar chart will be displayed.
6. Select the required subtype.
7. The chart will be placed in the current worksheet. The chart can be moved to any location in the worksheet.



Charts group in the Insert Tab

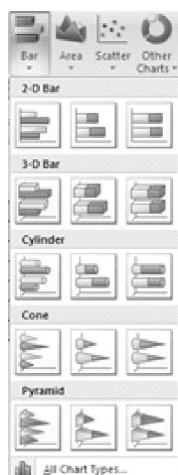
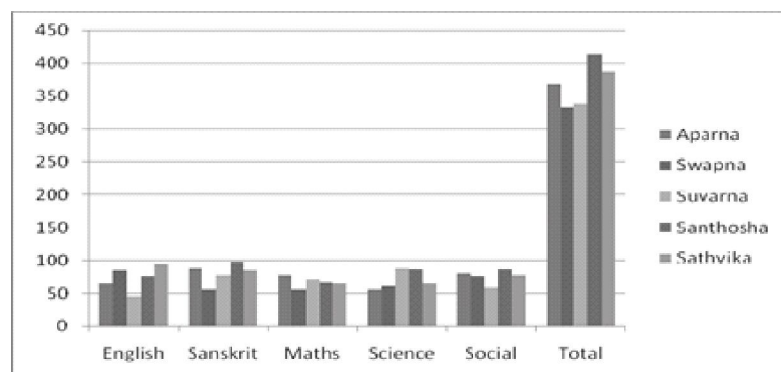


Chart Subtype

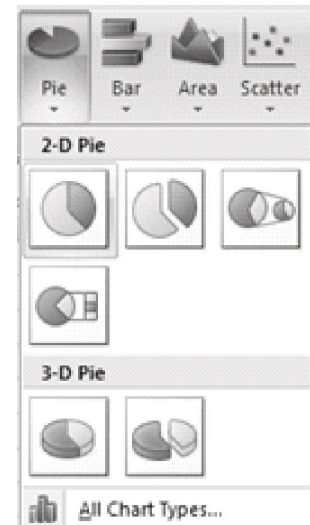


Chart

Note : If the data is not continuous, i.e., if the data is not present in successive cells, press Ctrl key for selecting the non - continuous data before creating a chart.

To create a Pie chart,

1. Click the Insert tab.
2. On the Charts group, click on the Pie chart icon.
All the subtypes of Pie chart will be displayed.
3. Select the required subtype. A blank chart appears (Note that we have not selected any cells in the beginning).
You can specify the data range at this point of time.
4. Select the blank chart and right click your mouse.
5. Choose the option 'Select Data'. (Alternatively you can click the Select Data icon from the Design tab)



- The Select Data Source dialog box will be displayed.
6. Click the Data Source button from the Chart data range text box.
The Select Data Source dialog box will minimize automatically.
 7. Select the cells containing the data range.
Data range gets filled in the Chart data range box.
 8. Click the Data Source button to view the dialog box.
 9. Click the OK button.

Modifying Charts

Once a chart is created, it can also be modified further.

Changing the title

To change the chart title, click the chart title, delete the existing title and type a new title.

Changing the source data

If the source data is changed, the chart need not be created again. The changes will be reflected automatically in the charts.

To change the source data:

1. Right-click on the chart.
2. On the shortcut menu, click Source Data.

The Source Data dialog box is displayed.

3. Click the Data Range button.
4. Select the range of cells that should appear in the chart.
5. Click the data range button again.
6. Click the OK button.

Changing the chart type :

Even after the chart is created, the chart type can be changed later on. To change the chart type.

1. Right-click the chart area.
2. Choose Change Chart Type.
3. Alternatively you can click the Change Chart type icon from the Design tab.
4. In the Change Chart Type dialog box, choose any other chart type
5. Click the OK button.

Formatting the chart

Once the chart is completed, the size, colour and the font of various elements in the chart can be changed.

a. To format the chart area,

1. Right-click on the chart area.
2. Choose Format Chart Area.
3. The Format Chart Area dialog box will be displayed.
4. The left side pane that is displayed has various options using which the appearance of the chart can be changed.

b. To change the border colour,

1. Click the Custom button.

2. Choose the colour from the colour options.
3. Select the Solid line option.
4. Click the Color icon.
5. A list of colours will be displayed.
6. Select the colour.
7. Click the Border Color button.

c. To make the border thicker,

1. Click the Border Styles button.
2. Increase the width to the required points by using the spin arrows.

d. To fill the chart area with different colour.

1. Click the Fill button.
2. Choose the Solid fill option.
3. Click the Color icon.
4. Select the colour from the colour box.
5. Click the Close button.

Printing Charts

You can print charts in the same way as you print worksheets.

Summary

- Charts represent data in a worksheet in the form of a picture.
- You can create charts by using the *Chart group* on the *Insert Tab*.



Teacher Activity

Demonstrate with example how to create a chart, format it and print the chart.



Student Activity

1. Open the Marks.xlsx file.
2. Create an embedded column chart that has the student names in the X-Axis and the percentage in the Y-Axis.

3. Change the mark of any student and observe the change in the chart.
4. Change the chart to a bar chart.
5. Remove the last student's data from the chart by changing the data range.



Test Your Knowledge

True or False

1. Charts are pictures that represent the data in a worksheet.
2. The *Chart* option is available on the *Chart* group.
3. The chart type cannot be changed.
4. If you change the data of the source cells, the chart will not reflect the changes.
5. Embedded charts cannot be printed.

Learning Objectives

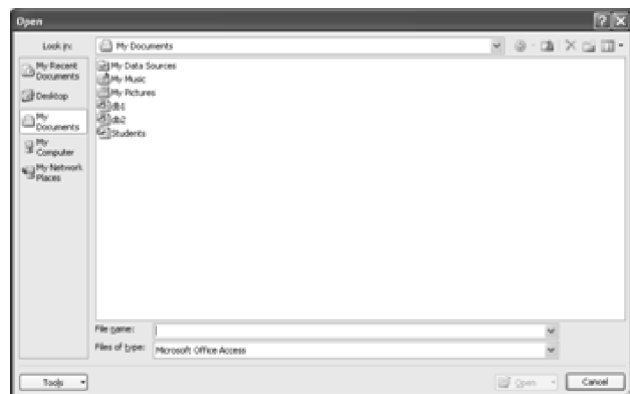
In this chapter you will be able to understand about ;

- Create a table by using the Table Templates
- Create a table by using the Design View
- Create a table by using the Datasheet View
- Set primary key
- Save a table

Opening a Database

You can open an existing database by using the following steps.

1. Click the Microsoft Office button and then click Open.
2. In the Open Dialog Box, browse the database that you want to open.
3. Select the required database.
4. Click the *Open* button.



The Open dialog box

Table Template

In the previous versions of MS Office Access, Access Table Wizard was used to quickly create a table from sample tables and fields. In Office Access 2007, table templates and field templates can be used to create a table.

A table template is an empty table that can be used as it is or modified to suit the needs.

In Office Access 2007, five table templates are available. They are:

- Contacts
- Tasks
- Issues
- Events
- Assets

- **Contacts:** A table for managing business contact information, which includes e-mail addresses, Web page URLs and so on
- **Tasks:** A table for tracking tasks that includes a field for attachments
- **Issues:** A table for tracking issues, which includes a field for attachments. It also has an append-only Memo field that keeps a history of old field values
- **Events:** A table for managing events
- **Assets:** A table for managing business assets

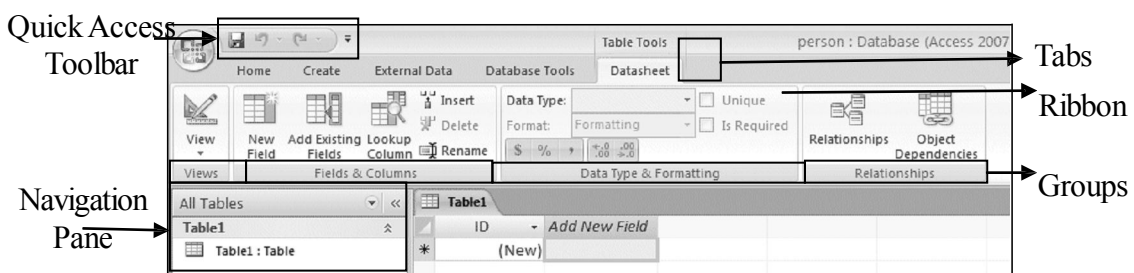
After a table is created by using a table template, fields may be added using field templates. A field template is a predefined field that can be added to any table in the Datasheet view.

Create a New Table by Using a Table Template

Steps to create a new table by using a table template,

1. Click the Microsoft Office Button and then click Open.
2. In the Open dialog box, select and open the database in which a table has to be created.
3. On the Create tab, in the Tables group, click Table Templates and then select one of the available templates from the list.
4. A new table is inserted, based on the table template that is chosen.

The components of the table window are shown in the following figure.



Components of the table window

Creating a Table using the Design View

To create a new table in a new database,

1. Click the Microsoft Office button and then click New.
2. In the File Name box, type a file name. To change the location, click the folder icon to browse.

3. Click Create.
4. The new database is opened and a new table named Table1 is created and opened in Datasheet view.

To switch to Design view,

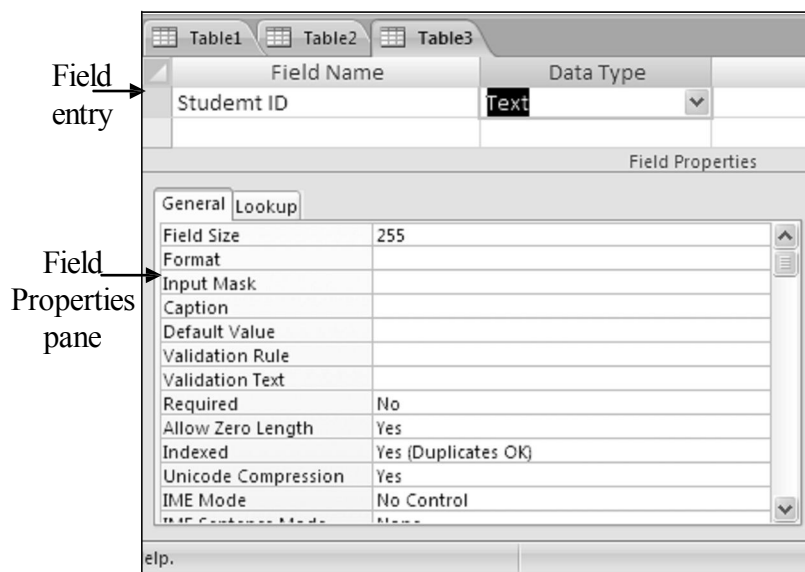
1. Right-click the table name in the Navigation Pane and then click Design View.

Or

2. Right-click the table name in the Table's Document tab and then click Design View.

Or

3. Click the Design View option on right bottom corner of the Status bar.
4. It will prompt the user to save the existing table to switch into Design View. To create a new table in an existing database,



The Design view of the table

1. Click the Microsoft Office Button and then click Open.
2. In the Open dialog box, select the database that you want to open and then click Open.
3. On the Create tab, in the Tables group, click Table Design.
4. A new table is inserted in the database and the table opens in Design View.

Similarly, you can create Player Details table using the Design view option in the New Table dialog box.

The Design view of the table consists of two parts:

- The Field entry pane – Is used for entering the field names, data type and their description.
- The Field Properties pane – Is used for setting the required properties for each of the fields.

The *Field Properties* pane is used to set the properties for each of the fields in the *Field Entry* pane. Some of the important options of the *Field Properties* pane are as follows:

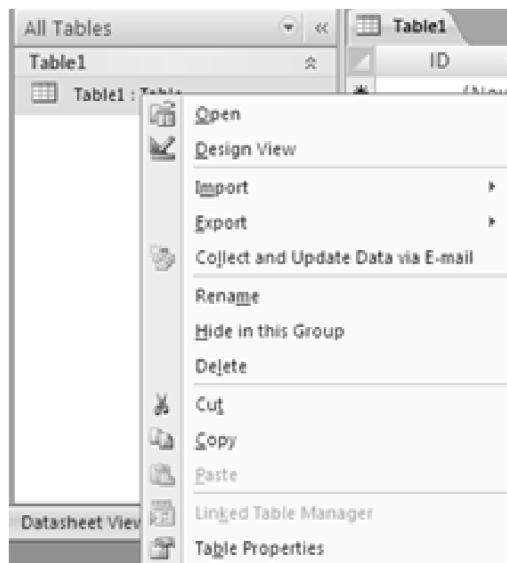
- **Field Size** – It determines the number of characters or digits that can be stored in a field.
- **Format** – It is used to specify the format in which a particular field value has to be displayed. The format varies for different data types.
- **Input Mask** – It allows you to control the user input. It consists of literal characters such as brackets, periods or hyphens and mask characters, which specify the valid characters allowed to be stored in the field.
- **Caption** – It can be used to display alternate names for fields in forms and reports.
- **Default Value** – It is used to specify a value that will be automatically displayed in the field. This value can be changed during data entry, if required.
- **Validation Rule** – It is used to specify a condition for the input. Only those values that follow the given condition will be accepted.
- **Validation Text** – It is used to display an error message if the Validation Rule is not satisfied.
- **Required** – This property when set to Yes ensures that the user does not skip an entry for the field during data entry.
- **Allow Zero Length** – This property when set to Yes denotes that the field can be left blank.
- **Indexed** – An index is used for faster data retrieval. Indexed property indicates whether the table data should be indexed based on the specific field. The fields, whose data are retrieved often, can be indexed to facilitate faster retrieval of records.

5. Enter the data in the *Field entry* and *Field Properties* areas.

The table is created in the Design view.

Tip : The *Tab* key is used to switch between the Field Name, Data Type and Description.

Note : To view the table in Design view, right click the table in the Navigation Pane, select Design View.



Design view option from the Navigation Pane

Setting the Primary Key

Before you set a field as the primary key of a table, you need to ensure that the field uniquely identifies each record stored in the table.

The Primary key is a column or combination of columns whose values uniquely identify a row or record in the table. The Primary key field must be selected based on the condition that the concerned field will have a unique data.

The primary key(s) will have a unique value for each record or row in the table. The characteristics of a perfect primary key are:

- It uniquely identifies each row.
- There is always a value, that is, it is never empty or null.
- It rarely changes.

Access uses primary key fields to quickly bring data together from multiple tables.

Often, there is a unique identification number, such as an ID number, a serial number or a code, that serves as a primary key. For example, in a Customers table, each customer has a unique customer ID number. The customer ID field is the primary key of the table.

A primary key for a table should always be specified. Access automatically creates an index for the primary key, which helps to speed up queries and other operations. Access also ensures that every record has a value in the primary key field and that it is always unique.

When a new table created is in Datasheet View, Access automatically creates a primary key and assigns it a field name of ID and the AutoNumber data type. The field is hidden in Datasheet View, but it can be seen if switched to Design View.

To Set the Primary Key,

1. Open the table in Design view.
2. Select the field or fields that need to be used as the primary key.
3. To select one field, click the row selector for the field you want.
4. To select more than one field, hold down the Ctrl key and then click the row selector for each field.
5. On the Design tab, in the Tools group, click Primary Key.

A key indicator is added to the left of the field or fields that is specified as the primary key.

Tip : Alternatively, to set the primary key

1. Select the field to be set as primary key.
2. Select and right click the field.
3. Select the Primary Key option from the shortcut- menu displayed.

Saving a Table

To save a table:

1. Click the Microsoft Office Button and then click Save. (or)
2. Right-click the table's document tab and then click Save on the shortcut menu. (or)



3. Click Save on the Quick Access Toolbar.

Caution

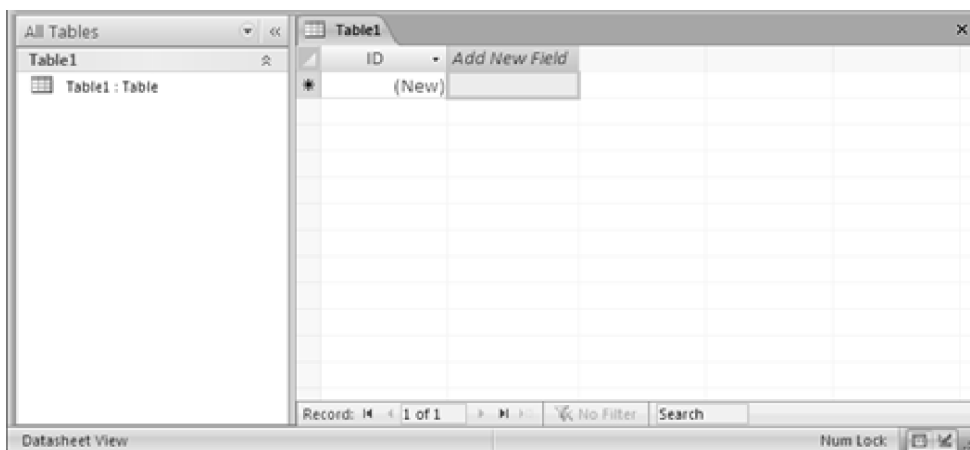
If you try to save a table without setting the primary key, Access displays a warning message asking you whether the table needs a primary key or not. If you click the Yes button, Access automatically creates an autonumber field as the primary key.

Creating a Table Using the Datasheet View

You can also create a table by using another option called the Datasheet View. In the Datasheet View, you cannot set the data type of a field, change the properties of a field or set a primary key.

To create a new table in a new database by using Datasheet View,

1. Click the Microsoft Office Button and then click New.
2. In the File Name box, type a file name for the new database.
3. To save the database in a different location, click the folder icon.
4. Click Create.
5. The new database opens and a new table named Table1 is created and opened in Datasheet view.



The datasheet view

To create a new table in an existing database by using Datasheet View,

1. Click the Microsoft Office Button and then click Open.
2. In the Open dialog box, select the database that needs to be opened and then click Open.

3. On the Create tab, in the Tables group, click Table.
4. A new table is inserted in the database and the table opens in Datasheet view.

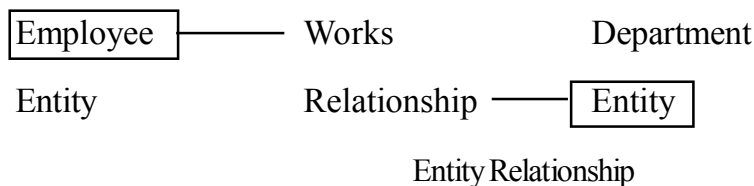
Extracting data across tables

Access allows you to retrieve data across tables. When related data are distributed across tables, you have to establish relationship among tables to retrieve data. For example, if you have a database on cricket and have different tables where you save the details of all player. You might want to retrieve the details of players playing for the South African team you need to relate Player Details and Team Details tables.

Types of Relationship

Relationship is a vital part in designing a database. The relationship between tables explains, how each table is related with the other and their dependent values. It is used to establish a connection between a pair of logical related entities.

For example, consider the relationship between an employee and department. Here, the entities are *Employee* and *Department*. An employee works in a particular department and a single department can have various employees. The relationship between them is *Works*. The relationship explained above is represented as follows.



The three types of relationships are:

- One-to-one (1:1) relationship
- One-to-many (1: m) or Many- to-one (m:1)
- Many-to-many (m:m)

Definition of Normalization

Normalization is a scientific method of breaking down complex table structures into simple table structures by using certain rules. Using this method, you can, reduce redundancy in a table and eliminate the problems of inconsistency and disk space usage. You can also ensure that there is no loss of information.

Summary :

- The five table templates in Office Access 2007 are Contacts, Tasks, Issues, Events and Assets
- The Design View option can be used to create a table to suit the needs
- A Datasheet View is mainly used to enter values in fields created in the Design View
- The Primary key is a column or combination of columns whose values uniquely identify a row or record in the table
- The Primary key field must be selected based on the condition that the concerned field will have a unique data
- If a table created by using the Table Design option is saved without setting a primary key, Microsoft Access displays a warning message asking whether the table needs a primary key or not. If the Yes button is clicked, Microsoft Access automatically creates an auto number field as the primary key



Teacher Activity

Demonstrate creation of data sheet.



Student Activity

1. Open the Cricket database.
2. Create Player Details table in design view. The table should contain the following fields:

Player ID, Player Name, Runs Scored, Wickets Taken, No of Centuries and No of Matches Played.
3. Set the Player ID field as the primary key.
4. Create Team Details table in Datasheet view. The table should contain the following fields: Team ID, Team Name, Coach Name and Overall Ranking.
5. Set the Team ID field as the primary key.
6. Open the Player Details table in datasheet view and enter the following data.

<i>Player ID</i>	<i>Player Name</i>	<i>Runs Scored</i>	<i>Wickets Taken</i>	<i>No of Centuries</i>	<i>No of Matches Played</i>
P001	Sachin Tendulkar	11000	106	33	297
P002	Michael Bevan	5400	70	6	101
P003	MuthiahMuralidharan	300	302	0	210
P004	Glenn Mcgrath	250	251	0	178
P005	Saurav Ganguly	6300	57	19	205
P006	Nasir Hussain	5855	45	1	170
P007	Jacques Kallis	5673	100	8	130
P008	Wasim Akram	3000	420	0	350
P009	Anil Kumble	560	250	0	200
P010	Ricky Ponting	6547	10	10	210

7. Enter the following data in the Team Details table.

<i>Team ID</i>	<i>Team Name</i>	<i>Coach Name</i>	<i>Overall Ranking</i>
T001	India	John Wright	5
T002	Australia	John Buchanan	1
T003	England	Duncan Fletcher	6
T004	Sri Lanka	Dave Whatmore	3
T005	South Africa	Eric Simons	2
T006	Pakistan	Richard Pybus	4
T007	New Zealand	Enis Eberhart	7



Test Your Knowledge

Fill in the blanks.

1. The Table option on the _____ tab is used to create the New Table.
2. To create the table in Design View, the Design View is selected from the _____ group.
3. In the Datasheet View, you cannot set the data type of a field. **(True/false)**
4. _____ property of the Design view is used to set the field length.
5. The Primary Key option is available on the _____ tab and the _____ group.

Introduction to Object-Oriented Programming**Introduction**

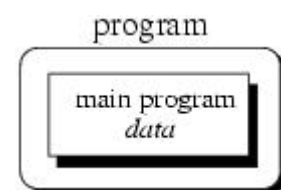
This lesson is a collection of lectures to be held in the on-line course *Introduction to Object-Oriented Programming Using C++*. In this course, object-orientation is introduced as a new programming concept which should help you in developing high quality software. Object-orientation is also introduced as a concept which makes developing of projects easier. However, this is **not** a course for learning the C++ programming language. If you are interested in learning the language itself, you might want to go through other tutorials, such as *C++: Annotations* by Frank Brokken and Karel Kubat. In this tutorial only those language concepts that are needed to present coding examples are introduced. And what makes object-orientation such a hot topic? To be honest, not everything that is sold under the term of object-orientation is really new. For example, there are programs written in procedural languages like Pascal or C which use object-oriented concepts. But there exist a few important features which these languages won't handle or won't handle very well, respectively.

Some people will say that object-orientation is “modern”. When reading announcements of new products everything seems to be “object-oriented”. “Objects” are everywhere. In this tutorial we will try to outline characteristics of object-orientation to allow you to judge those object-oriented products.

Unstructured Programming

Usually, people start learning programming by writing small and simple programs consisting only of one main program. Here “main program” stands for a sequence of commands or *statements* which modify data which is *global* throughout the whole program. We can illustrate this as shown in Fig.

Unstructured programming. The main program directly operates on global data.



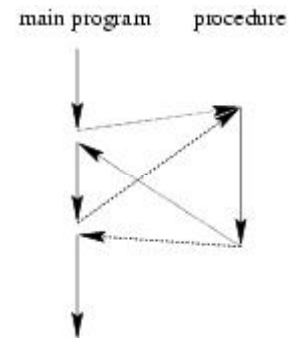
As you should all know, this programming techniques provide tremendous disadvantages once the program gets sufficiently large. For example, if the

same statement sequence is needed at different locations within the program, the sequence must be copied. This has led to the idea to *extract* these sequences, name them and offering a technique to call and return from these *procedures*.

Procedural Programming

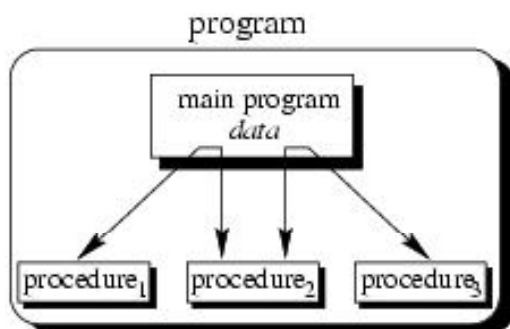
With procedural programming you are able to combine returning sequences of statements into one single place. A *procedure call* is used to invoke the procedure. After the sequence is processed, flow of control proceeds right after the position where the call was made.

Execution of procedures. After processing flow of controls proceed where the call was made.



With introducing *parameters* as well as procedures of procedures (*subprocedures*) programs can now be written more structured and error free. For example, if a procedure is correct, every time it is used it produces correct results. Consequently, in cases of errors you can narrow your search to those places which are not proven to be correct.

Now a program can be viewed as a sequence of procedure calls. The main program is responsible to pass data to the individual calls, the data is processed by the procedures and, once the program has finished, the resulting data is presented. Thus, the *flow of data* can be illustrated as a hierarchical graph, a *tree*, as shown in Fig. the below for a program with no sub procedures.



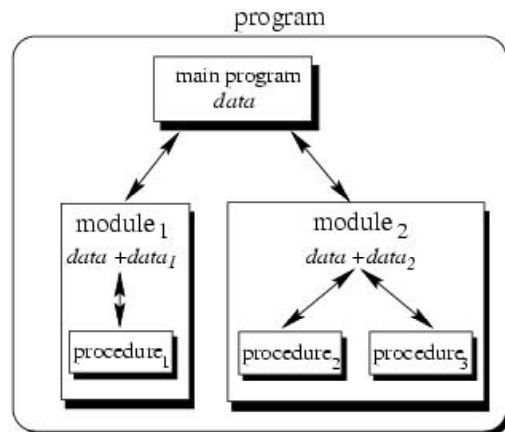
Procedural programming. The main program coordinates calls to procedures and hands over appropriate data as parameters.

To sum up: Now we have a single program which is divided into small pieces called procedures. To enable usage of general procedures or groups of procedures also in other programs, they must

be separately available. For that reason, modular programming allows grouping of procedures into modules.

Modular Programming

With modular programming procedures of a common functionality are grouped together into separate *modules*. A program therefore no longer consists of only one single part. It is now divided into several smaller parts which interact through procedure calls and which form the whole program.



Modular programming. The main program coordinates calls to procedures in separate modules and hands over appropriate data as parameters.

Each module can have its own data. This allows each module to manage an internal *state* which is modified by calls to procedures of this module. However, there is only one state per module and each module exists at most once in the whole program.

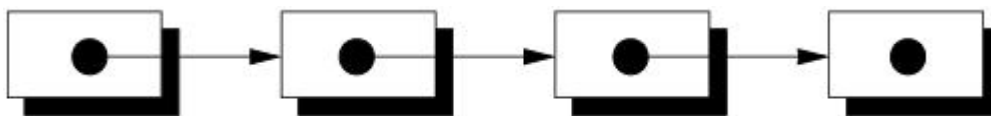
An Example with Data Structures

Programs use data structures to store data. Several data structures exist, for example lists, trees, arrays, sets, bags or queues to name a few. Each of these data structures can be characterized by their *structure* and their *access methods*.

Handling Single Lists

You all know singly linked lists which use a very simple structure, consisting of elements which are strung together, as shown in below picture.

Structure of a singly linked list.



Singly linked lists just provides access methods to *append* a new element to their end and to *delete* the element at the front. Complex data structures might use already existing ones. For example a *queue* can be structured like a singly linked list. However, queues provide access methods to *put* a data element at the end and to *get* the first data element (*first-in first-out* (FIFO) behaviour).

We will now present an example which we use to present some design concepts. Since this example is just used to illustrate these concepts and problems it is neither complete nor optimal. Refer to chapter 10 for a complete object-oriented discussion about the design of data structures.

Suppose you want to program a list in a modular programming language such as C or Modula-2. As you believe that lists are a common data structure, you decide to implement it in a separate *module*. Typically, this requires you to write two files: the *interface definition* and the *implementation file*. Within this chapter we will use a very simple pseudo code which you should understand immediately. Let's assume, that comments are enclosed in “/* ... */”. Our interface definition might then look similar to that below:

```
/*  
 * Interface definition for a module which implements  
 * a singly linked list for storing data of any type.  
 */
```

```
MODULE Singly-Linked-List-1  
  BOOL list_initialize();  
  BOOL list_append(ANY data);  
  BOOL list_delete(); list_end();  
  ANY list_getFirst();  
  ANY list_getNext();  
  BOOL list_isEmpty();  
END Singly-Linked-List-1
```

Interface definitions just describe *what* is available and not *how* it is made available. You *hide* the information of the implementation in the implementation file. This is a fundamental principle in software engineering, so let's repeat it: You *hide* information of the actual implementation (*information hiding*). This enables you to change the implementation, for example to use a faster but more memory consuming algorithm for storing elements without the need to change other modules of your program: The calls to provided procedures remain the same.

The idea of this interface is as follows: Before using the list one has to call *list_initialize()* to initialize variables local to the module. The following two procedures implement the mentioned access methods *append* and *delete*. The *append* procedure needs a more detailed discussion. Function *list_append()* takes one argument *data* of arbitrary type. This is necessary since you wish to use your list in several different environments, hence, the type of the data elements to be stored in the list is not known beforehand. Consequently, you have to use a special type *ANY* which allows to assign data of any type to it. The third procedure *list_end()* needs to be called when the program terminates to enable the module to clean up its internally used variables. For example you might want to release allocated memory.

With the next two procedures *list_getFirst()* and *list_getNext()* a simple mechanism to traverse through the list is offered. Traversing can be done using the following loop:

```
ANY data;
```

```
data <- list_getFirst();
```

```
WHILE data IS VALID DO
```

```
doSomething(data);
```

```
data <- list_getNext();
```

```
END
```

Now you have a list module which allows you to use a list with any type of data elements. But what, if you need more than one list in one of your programs?

Handling Multiple Lists

You decide to redesign your list module to be able to manage more than one list. You therefore create a new interface description which now includes a definition for a *list handle*. This handle is used in every provided procedure to uniquely identify the list in question. Your interface definition file of your new list module looks like this:

```
/*
```

```
* A list module for more than one list.
```

```
*/
```

```

MODULE Singly-Linked-List-2

DECLARE TYPE list_handle_t;

list_handle_t list_create();
    list_destroy(list_handle_t this);
BOOL list_append(list_handle_t this, ANY data);
ANY list_getFirst(list_handle_t this);
ANY list_getNext(list_handle_t this);
BOOL list_isEmpty(list_handle_t this);

END Singly-Linked-List-2;

```

You use *DECLARE TYPE* to introduce a new type *list_handle_t* which represents your list handle. We do not specify, how this handle is actually represented or even implemented. You also *hide* the implementation details of this type in your implementation file. Note the difference to the previous version where you just hide functions or procedures, respectively. Now you also hide information for an user defined data type called *list_handle_t*.

You use *list_create()* to obtain a handle to a new thus empty list. Every other procedure now contains the special parameter *this* which just identifies the list in question. All procedures now operate on this handle rather than a module global list.

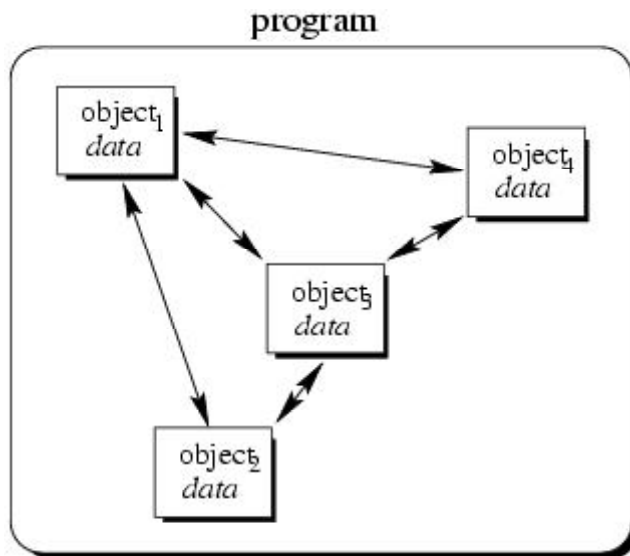
Now you might say, that you can create *list objects*. Each such object can be uniquely identified by its handle and only those *methods* are applicable which are defined to operate on this handle.

Object-Oriented Programming

Object-oriented programming solves some of the problems just mentioned. In contrast to the other techniques, we now have a web of interacting *objects*, each house-keeping its own state as shown in below diagram.

Object-oriented programming. Objects of the program interact by sending messages to each other.

Consider the multiple lists example again. The problem here with modular programming is, that you must explicitly create and destroy your list handles. Then you use the procedures of the module to modify each of your handles.



In contrast to that, in object-oriented programming we would have as many list objects as needed. Instead of calling a procedure which we must provide with the correct list handle, we would directly send a message to the list object in question. Roughly speaking, each object implements its own module allowing for example many lists to coexist.

Each object is responsible to initialize and destroy itself correctly. Consequently, there is no longer the need to explicitly call a creation or termination procedure.

You might ask: *So what? Isn't this just a more fancier modular programming technique?* You were right, if this would be all about object-orientation. Fortunately, it is not. Beginning with the next chapters additional features of object-orientation are introduced which makes object-oriented programming to a new programming technique.

Properties of Abstract Data Types

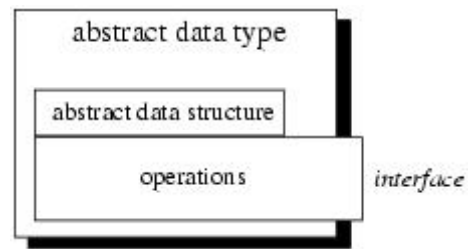
The example of the previous section shows, that with abstraction you create a well-defined entity which can be properly handled. These entities define the *data structure* of a set of items. For example, each administered employee has a name, date of birth and social number.

The data structure can only be accessed with defined *operations*. This set of operations is called *interface* and is *exported* by the entity. An entity with the properties just described is called an *abstract data type* (ADT).

An ADT which consists of an abstract data structure and operations. Only the operations are viewable from the outside and define the interface.

Abstract data type (ADT).

Once a new employee is “created” the data structure is filled with actual values: You now have an *instance* of an abstract employee. You can create as many instances of an abstract employee as needed to describe every real employed person.



Let’s try to put the characteristics of an ADT in a more formal way:

Definition (Abstract Data Type) An *abstract data type (ADT)* is characterized by the following properties:

1. It exports a **type**.
2. It exports a **set of operations**. This set is called **interface**.
3. Operations of the interface are the one and only access mechanism to the type’s data structure.
4. Axioms and preconditions define the application domain of the type.

With the first property it is possible to create more than one instance of an ADT as exemplified with the employee example. You might also remember the list example of chapter 2. In the first version we have implemented a list as a module and were only able to use one list at a time. The second version introduces the “handle” as a reference to a “list object”. From what we have learned now, the handle in conjunction with the operations defined in the list module defines an ADT *List*:

1. When we use the handle we define the corresponding variable to be of type *List*.
2. The interface to instances of type *List* is defined by the interface definition file.
3. Since the interface definition file does not include the actual representation of the handle, it cannot be modified directly.
4. The application domain is defined by the semantical meaning of provided operations. Axioms and preconditions include statements such as
 - “An empty list is a list.”
 - “Let $l=(d1, d2, d3, \dots, dN)$ be a list. Then $l.append(dM)$ results in $l=(d1, d2, d3, \dots, dN, dM)$.”

- “The first element of a list can only be deleted if the list is not empty.”

However, all of these properties are only valid due to our understanding of and our discipline in using the list module. It is in our responsibility to use instances of *List* according to these rules.

Importance of Data Structure Encapsulation

The principle of hiding the used data structure and to only provide a well-defined interface is known as *encapsulation*. Why is it so important to encapsulate the data structure?

To answer this question consider the following mathematical example where we want to define an ADT for complex numbers. For the following it is enough to know that complex numbers consists of two parts: *real part* and *imaginary part*. Both parts are represented by real numbers. Complex numbers define several operations: addition, subtraction, multiplication or division to name a few. Axioms and preconditions are valid as defined by the mathematical definition of complex numbers. For example, it exists a neutral element for addition.

To represent a complex number it is necessary to define the data structure to be used by its ADT. One can think of at least two possibilities to do this:

- Both parts are stored in a two-valued array where the first value indicates the real part and the second value the imaginary part of the complex number. If x denotes the real part and y the imaginary part, you could think of accessing them via array subscription: $x=c[0]$ and $y=c[1]$.
- Both parts are stored in a two-valued record. If the element name of the real part is r and that of the imaginary part is i , x and y can be obtained with: $x=c.r$ and $y=c.i$.

Point 3 of the ADT definition says that for each access to the data structure there must be an operation defined. The above access examples seem to contradict this requirement. Is this really true?

Let's look again at the two possibilities for representing imaginary numbers. Let's stick to the real part. In the first version, x equals $c[0]$. In the second version, x equals $c.r$. In both cases x equals “something”. It is this “something” which differs from the actual data structure used. But in both cases the performed operation “equal” has the same meaning to declare x to be equal to the real part of the complex number c : both cases achieve the same semantics.

If you think of more complex operations the impact of decoupling data structures from operations becomes even more clear. For example the addition of two complex numbers requires you to perform an addition for each part. Consequently, you must access the value of each part which is different for each version. By providing an operation “add” you can *encapsulate* these details from its actual use. In an application context you simply “add two complex numbers” regardless of how this functionality is actually achieved.

Once you have created an ADT for complex numbers, say *Complex*, you can use it in the same way like well-known data types such as integers.

Let’s summarize this: The separation of data structures and operations and the constraint to only access the data structure via a well-defined interface allows you to choose data structures appropriate for the application environment.

Generic Abstract Data Types

ADTs are used to define a new type from which instances can be created. As shown in the list example, sometimes these instances should operate on other data types as well. For instance, one can think of lists of apples, cars or even lists. The semantical definition of a list is always the same. Only the type of the data elements change according to what type the list should operate on.

This additional information could be specified by a *generic parameter* which is specified at instance creation time. Thus an instance of a *generic ADT* is actually an instance of a particular variant of the ADT. A list of apples can therefore be declared as follows:

```
List<Apple> listOfApples;
```

The angle brackets now enclose the data type for which a variant of the generic ADT *List* should be created. *listOfApples* offers the same interface as any other list, but operates on instances of type *Apple*.

Notation :

As ADTs provide an abstract view to describe properties of sets of entities, their use is independent from a particular programming language. Each ADT description consists of two parts:

- **Data:** This part describes the structure of the data used in the ADT in an informal way.

- **Operations:** This part describes valid operations for this ADT, hence, it describes its interface. We use the special operation **constructor** to describe the actions which are to be performed once an entity of this ADT is created and **destructor** to describe the actions which are to be performed once an entity is destroyed. For each operation the provided *arguments* as well as *preconditions* and *post conditions* are given.

As an example the description of the ADT *Integer* is presented. Let k be an integer expression:

ADT *Integer* is Data

A sequence of digits optionally prefixed by a plus or minus sign. We refer to this signed whole number as N .

Operations

constructor

Creates a new integer.

add(k)

Creates a new integer which is the sum of N and k .

Consequently, the *postcondition* of this operation is $sum = N+k$. Don't confuse this with assign statements as used in programming languages! It is rather a mathematical equation which yields "true" for each value sum , N and k after *add* has been performed.

sub(k)

Similar to *add*, this operation creates a new integer of the difference of both integer values. Therefore the postcondition for this operation is $sum = N-k$.

set(k)

Set N to k . The postcondition for this operation is $N = k$.

...

end

The description above is a *specification* for the ADT *Integer*. Please notice, that we use words for names of operations such as "add". We could use the more intuitive "+" sign instead, but this may lead to some confusion: You must distinguish the operation "+" from the mathematical use of "+" in the postcondition. The name of the operation is just *syntax* whereas

the *semantics* is described by the associated pre- and postconditions. However, it is always a good idea to combine both to make reading of ADT specifications easier.

Real programming languages are free to choose an arbitrary *implementation* for an ADT. For example, they might implement the operation *add* with the infix operator “+” leading to a more intuitive look for addition of integers.

Abstract Data Types and Object-Oriented

ADTs allows the creation of instances with well-defined properties and behaviour. In object-orientation ADTs are referred to as *classes*. Therefore a class defines properties of *objects* which are the instances in an object-oriented environment.

ADTs define functionality by putting main emphasis on the involved data, their structure, operations as well as axioms and preconditions. Consequently, object-oriented programming is “programming with ADTs”: combining functionality of different ADTs to solve a problem. Therefore instances (objects) of ADTs (classes) are dynamically created, destroyed and used.

More Object-Oriented Concepts

Peter Müller Globewide Network Academy (GNA) pmueller@uu-gna.mit.edu
Whereas the previous lecture introduces the fundamental concepts of object-oriented programming, this lecture presents more details about the object-oriented idea. This section is mainly adopted from [2].

Relationships

In exercise 3.6.5 you already investigate relationships between abstract data types and instances and describe them in your own words. Let’s go in more detail here.

A-Kind-Of relationship

Consider you have to write a drawing program. This program would allow drawing of various *objects* such as points, circles, rectangles, triangles and many more. For each object you provide a *class* definition. For example, the point class just defines a point by its coordinates:

```
class Point {  
  attributes:  
  int x, y
```

methods:

```
setX(int newX)
```

```
getX()
```

```
setY(int newY)
```

```
getY()
```

```
}
```

You continue defining classes of your drawing program with a class to describe circles. A circle defines a center point and a radius:

```
class Circle {
```

attributes:

```
int x, y,
```

```
radius
```

methods:

```
setX(int newX)
```

```
getX()
```

```
setY(int newY)
```

```
getY()
```

```
setRadius(newRadius)
```

```
getRadius()
```

```
}
```

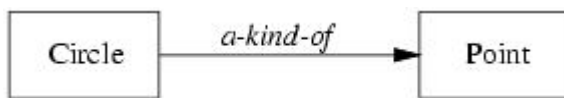
Comparing both class definitions we can observe the following:

- Both classes have two data elements x and y . In the class *Point* these elements describe the position of the point, in the case of class *Circle* they describe the circle's center. Thus, x and y have the same meaning in both classes: They describe the position of their associated object by defining a point.

- Both classes offer the same set of methods to get and set the value of the two data elements x and y .
- Class *Circle* “adds” a new data element *radius* and corresponding access methods.

Knowing the properties of class *Point* we can describe a circle as a point plus a radius and methods to access it. Thus, a circle is “a-kind-of” point. However, a circle is somewhat more “specialized”. We illustrate this graphically as shown below.

Illustration of “a-kind-of” relationship.



In this and the following figures, classes are drawn using rectangles. Their name always starts with an uppercase letter. The arrowed line indicates the direction of the relation, hence, it is to be read as “Circle is a-kind-of Point.”

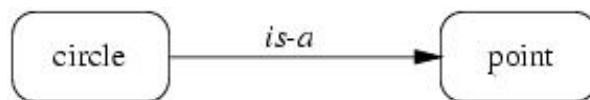
Is-A relationship

The previous relationship is used at the class level to describe relationships between two similar classes. If we create objects of two such classes we refer to their relationship as an “is-a” relationship.

Since the class *Circle* is a kind of class *Point*, an instance of *Circle*, say *acircle*, is a *point*. Consequently, each circle behaves like a point. For example, you can move points in x direction by altering the value of x . Similarly, you move circles in this direction by altering their x value.

Below diagram illustrates this relationship. In this and the following figures, objects are drawn using rectangles with round corners. Their name only consists of lowercase letters.

Illustration of “is-a” relationship.



Part-Of relationship

You sometimes need to be able to build objects by combining them out of others. You already know this from procedural programming, where you have the structure or record construct to put data of various types together.

Let's come back to our drawing program. You already have created several classes for the available figures. Now you decide that you want to have a special figure which represents your own logo which consists of a circle and a triangle. (Let's assume, that you already have defined a class *Triangle*.) Thus, your logo consists of two *parts* or the circle and triangle are *part-of* your logo:

```
class Logo {
  attributes:
    Circle circle
    Triangle triangle

  methods:
    set(Point where)
}
```

We illustrate this as below.

Illustration of “part-of” relationship.



Has-A relationship

This relationship is just the inverse version of the part-of relationship. Therefore we can easily add this relationship to the part-of illustration by adding arrows in the other direction show in the below diagram.

Illustration of “has-a” relationship.



Inheritance

With inheritance we are able to make use of the a-kind-of and is-a relationship. As described there, classes which are a-kind-of another class share properties of the latter. In our point and circle example, we can define a circle which *inherits from* point:

```

class Circle inherits from Point {
attributes:
    int radius

methods:
    setRadius(int newRadius)
    getRadius()
}

```

Class *Circle* inherits all data elements and methods from point. There is no need to define them twice: We just use already existing and well-known data and method definitions.

On the object level we are now able to use a circle just as we would use a point, because a circle is-a point. For example, we can define a circle object and set its center point coordinates:

```

Circle acircle
acircle.setX(1)    /* Inherited from Point */
acircle.setY(2)
acircle.setRadius(3) /* Added by Circle */

```

“Is-a” also implies, that we can use a circle everywhere where a point is expected. For example, you can write a function or method, say *move()*, which should move a point in *x* direction:

```

move(Point apoint, int deltax) {
    apoint.setX(apoint.getX() + deltax)
}

```

As a circle inherits from a point, you can use this function with a circle argument to move its center point and, hence, the whole circle:

```

Circle acircle
...

```

```
move(acircle, 10) /* Move circle by moving */  
/* its center point */
```

Let's try to formalize the term “inheritance”:

Definition (Inheritance) *Inheritance is the mechanism which allows a class A to inherit properties of a class B. We say “A inherits from B”. Objects of class A thus have access to attributes and methods of class B without the need to redefine them. The following definition defines two terms with which we are able to refer to participating classes when they use inheritance.*

Definition (Superclass/Subclass) *If class A inherits from class B, then B is called superclass of A. A is called subclass of B. Objects of a subclass can be used where objects of the corresponding superclass are expected. This is due to the fact that objects of the subclass share the same behaviour as objects of the superclass.*

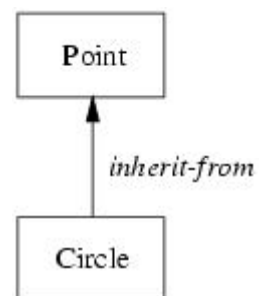
In the literature you may also find other terms for “superclass” and “subclass”. Superclasses are also called *parent classes*. Subclasses may also be called *child classes* or just *derived classes*.

Of course, you can again inherit from a subclass, making this class the superclass of the new subclass. This leads to a hierarchy of superclass/subclass relationships. If you draw this hierarchy you get an *inheritance graph*.

A common drawing scheme is to use arrowed lines to indicate the inheritance relationship between two classes or objects. In our examples we have used “inherits-from”. Consequently, the arrowed line starts from the subclass towards the superclass as illustrated in below diagram.

A simple inheritance graph.

In the literature you also find illustrations where the arrowed lines are used just the other way around. The direction in which the arrowed line is used, depends on how the corresponding author has decided to understand it.



Anyway, within this tutorial, the arrowed line is always directed towards the superclass.

In the following sections an unmarked arrowed line indicates “inherit-from”.

Multiple Inheritance

One important object-oriented mechanism is multiple inheritance. Multiple inheritance does **not** mean that multiple subclasses share the same superclass. It also does **not** mean that a subclass can inherit from a class which itself is a subclass of another class.

Multiple inheritance means that one subclass can have *more than one* superclass. This enables the subclass to inherit properties of more than one superclass and to “merge” their properties.

As an example consider again our drawing program. Suppose we already have a class *String* which allows convenient handling of text. For example, it might have a method to *append* other text. In our program we would like to use this class to add text to the possible drawing objects. It would be nice to also use already existing routines such as *move()* to move the text around. Consequently, it makes sense to let a drawable text have a point which defines its location within the drawing area. Therefore we derive a new class *DrawableString* which inherits properties from *Point* and *String* as illustrated in below diagram.

Derive a drawable string which inherits properties of *Point* and *String*.

In our pseudo language we write this by simply separating the multiple superclasses by comma:

```
class DrawableString inherits from Point, String {
```

```
attributes:
```

```
    /* All inherited from superclasses */
```

```
methods:
```

```
    /* All inherited from superclasses */
```

```
}
```

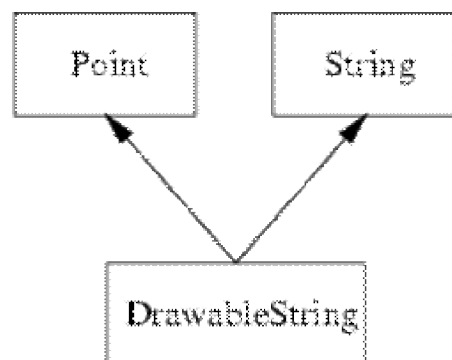
We can use objects of class *DrawableString* like both points and strings. Because a *drawablestring* is-a *point* we can move them around

```
DrawableString dstring
```

```
...
```

```
move(dstring, 10)
```

```
...
```



Since it is a *string*, we can append other text to them:

```
dstring.append("The red brown fox ...")
```

Now it's time for the definition of multiple inheritance:

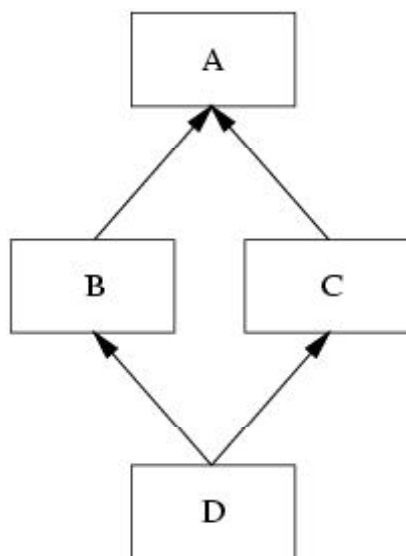
Definition (Multiple Inheritance) *If class A inherits from more than one class, ie. A inherits from B1, B2, ..., Bn, we speak of **multiple inheritance**. This may introduce **naming conflicts** in A if at least two of its superclasses define properties with the same name.*

The above definition introduce *naming conflicts* which occur if more than one superclass of a subclass use the same name for either attributes or methods. For an example, let's assume, that class *String* defines a method *setX()* which sets the string to a sequence of "X" characters. The question arises, what should be inherited by *DrawableString*? The *Point*, *String* version or none of them?

These conflicts can be solved in at least two ways:

- The order in which the superclasses are provided define which property will be accessible by the conflict causing name. Others will be "hidden".
- The subclass must resolve the conflict by providing a property with the name and by defining how to use the ones from its superclasses.

The first solution is not very convenient as it introduces implicit consequences depending on the order in which classes inherit from each other. For the second case, subclasses must explicitly redefine properties which are involved in a naming conflict.



A special type of naming conflict is introduced if a class *D* multiply inherits from superclasses *B* and *C* which themselves are derived from one superclass *A*. This leads to an inheritance graph as shown in below diagram.

A name conflict introduced by a shared superclass of superclasses used with multiple inheritance.

The question arises what properties class *D* actually inherits from its superclasses *B* and *C*. Some existing programming languages solve this special inheritance graph by deriving *D* with

- the properties of *A* plus
- the properties of *B* and *C* **without** the properties they have inherited from *A*.

Consequently, *D* cannot introduce naming conflicts with names of class *A*. However, if *B* and *C* add properties with the same name, *D* runs into a naming conflict.

Another possible solution is, that *D* inherits from both inheritance paths. In this solution, *D* owns **two** copies of the properties of *A*: one is inherited by *B* and one by *C*.

Although multiple inheritance is a powerful object-oriented mechanism the problems introduced with naming conflicts have lead several authors to “doom” it. As the result of multiple inheritance can always be achieved by using (simple) inheritance some object-oriented languages even don’t allow its use. However, carefully used, under some conditions multiple inheritance provides an efficient and elegant way of formulating things.

Polymorphism

Polymorphism allows an entity (for example, variable, function or object) to take a variety of representations. Therefore we have to distinguish different types of polymorphism which will be outlined here.

The first type is similar to the concept of dynamic binding. Here, the type of a variable depends on its content. Thus, its type depends on the content at a specific time:

```
v := 123    /* v is integer */
...        /* use v as integer */
v := 'abc'  /* v “switches” to string */
...        /* use v as string */
```

Definition (Polymorphism (1)) *The concept of dynamic binding allows a variable to take different types dependent on the content at a particular time. This ability of a variable is called **polymorphism**. Another type of polymorphism can be defined for functions. For example, suppose you want to define a function *isNull()* which returns TRUE if its argument is 0 (zero) and FALSE otherwise. For integer numbers this is easy:*

```
boolean isNull(int i) {
    if (i == 0) then
        return TRUE
```

```
else
    return FALSE
endif
}
```

However, if we want to check this for real numbers, we should use another comparison due to the precision problem:

```
boolean isNull(real r) {
    if (r < 0.01 and r > -0.99) then
        return TRUE
    else
        return FALSE
    endif
}
```

In both cases we want the function to have the name *isNull*. In programming languages without polymorphism for functions we cannot declare these two functions because the name *isNull* would be doubly defined. Without polymorphism for functions, doubly defined names would be ambiguous. However, if the language would take the *parameters* of the function into account it would work. Thus, functions (or methods) are uniquely identified by:

- the *name* of the function (or method) and
- the *types* of its *parameter list*.

Since the parameter list of both *isNull* functions differ, the compiler is able to figure out the correct function call by using the actual types of the arguments:

```
var i : integer
var r : real

i = 0
r = 0.0
```

...

```
if (isNull(i)) then ... /* Use isNull(int) */
```

...

```
if (isNull(r)) then ... /* Use isNull(real) */
```

Definition (Polymorphism (2)) *If a function (or method) is defined by the combination of*

- *its name and*
- *the list of types of its parameters*

*we speak of **polymorphism***. This type of polymorphism allows us to reuse the same name for functions (or methods) as long as the parameter list differs. Sometimes this type of polymorphism is called *overloading*.

The last type of polymorphism allows an object to choose correct methods. Consider the function *move()* again, which takes an object of class *Point* as its argument. We have used this function with any object of derived classes, because the is-a relation holds.

Now consider a function *display()* which should be used to display drawable objects. The declaration of this function might look like this:

```
display(DrawableObject o) {  
    ...  
    o.print()  
    ...  
}
```

We would like to use this function with objects of classes derived from *DrawableObject*:

Circle acircle

Point apoint

Rectangle arectangle

```
display(apoint) /* Should invoke apoint.print() */  
display(acircle) /* Should invoke acircle.print() */  
display(arectangle) /* Should invoke arectangle.print() */
```

The actual method should be defined by the *content* of the object *o* of function *display()*. Since this is somewhat complicated, here is a more abstract example:

```
class Base {  
  attributes:  
  
  methods:  
    virtual foo()  
    bar()  
}  
class Derived inherits from Base {  
  attributes:  
  
  methods:  
    virtual foo()  
    bar()  
}  
  
demo(Base o) {  
  o.foo()  
  o.bar()  
}
```

Base abase

Derived aderived

demo(abase)

demo(aderived)

In this example we define two classes *Base* and *Derived*. Each class defines two methods *foo()* and *bar()*. The first method is defined as virtual. This means that if this method is invoked its definition should be evaluated by the content of the object.

We then define a function *demo()* which takes a *Base* object as its argument. Consequently, we can use this function with objects of class *Derived* as the is-a relation holds. We call this function with a *Base* object and a *Derived* object, respectively.

Suppose, that *foo()* and *bar()* are defined to just print out their name and the class in which they are defined. Then the output is as follows:

foo() of Base called.

bar() of Base called.

foo() of Derived called.

bar() of Base called.

Why is this so? Let's see what happens. The first call to *demo()* uses a *Base* object. Thus, the function's argument is "filled" with an object of class *Base*. When it is time to invoke method *foo()* its actual functionality is chosen based on the current content of the corresponding object *o*. This time, it is a *Base* object. Consequently, *foo()* as defined in class *Base* is called.

The call to *bar()* is *not* subject to this content resolution. It is not marked as virtual. Consequently, *bar()* is called in the scope of class *Base*.

The second call to *demo()* takes a *Derived* object as its argument. Thus, the argument *o* is filled with a *Derived* object. However, *o* itself just represents the *Base* part of the provided object *aderived*.

Now, the call to *foo()* is evaluated by examining the content of *o*, hence, it is called within the scope of *Derived*. On the other hand, *bar()* is still evaluated within the scope of *Base*.

Definition (Polymorphism (3)) Objects of superclasses can be filled with objects of their subclasses. Operators and methods of subclasses can be defined to be evaluated in two contexts:

1. Based on object type, leading to an evaluation within the scope of the superclass.
2. Based on object content, leading to an evaluation within the scope of the contained subclass.

The second type is called **polymorphism**.

Introduction to C++

Peter Müller

Globewide Network Academy (GNA)

pmueller@uu-gna.mit.edu

This section is the first part of the introduction to C++. Here we focus on C from which C++ was adopted. C++ extends the C programming language with strong typing, some features and - most importantly - object-oriented concepts.

The C Programming Language

Developed in the late 1970s, C gained an huge success due to the development of UNIX which was almost entirely written in this language. In contrast to other high level languages, C was written by programmers for programmers. Thus it allows sometimes, say, weird things which in other languages such as Pascal are forbidden due to its bad influence on programming style. Anyway, when used with some discipline, C is as good a language as any other.

The comment in C is enclosed in `/* ... */`. Comments cannot be nested.

Data Types

Below shown table describes the built-in data types of C. The specified *Size* is measured in bytes on a 386 PC running Linux 1.2.13. The provided *Domain* is based on the *Size* value. You can obtain information about the size of a data type with the `sizeof` operator.

Built-in types.

Type	Description	Size	Domain
Char	Signed character/byte. Characters are enclosed in single quotes	1	-128..127
double	Double precision number	8	ca. 10^{-308} .. 10^{308}
int	Signed interer	4	-2^{31} .. $2^{31} - 1$
float	Floating point number	4	ca. 10^{-38} .. 10^{38}
long (int)	Signed long integer	4	-2^{31} .. $2^{31} - 1$
long long (int)	Signed very long integer	8	-2^{63} .. $2^{63} - 1$
Shot (int)	Short Integer	2	-2^{15} .. $2^{15} - 1$
unsigned char	Unsigned character/byte	1	0..255
unsigned (int)	Unsigned integer	4	$0..2^{32} - 1$
unsigned long (int)	Unsigned long integer	4	$0..2^{32} - 1$
unsigned long long (int)	Unsigned very long integer	8	$0..2^{64} - 1$
unsigned short (int)	Unsigned short integer	2	$0..2^{16} - 1$

Variables of these types are defined simply by preceding the name with the type:

```
int an_int;
```

```
float a_float;
```

```
long long a_very_long_integer;
```

With struct you can combine several different types together. In other languages this is sometimes called a *record*:

```
struct date_s {
    int day, month, year;
} aDate;
```

The above *definition* of aDate is also the *declaration* of a structure called date_s. We can define other variables of this type by referencing the structure by name:

```
struct date_s anotherDate;
```

We do not have to name structures. If we omit the name, we just cannot reuse it. However, if we name a structure, we can just *declare* it without defining a variable:

```
struct time_s {  
    int hour, minute, second;  
};
```

We are able to use this structure as shown for anotherDate. This is very similar to a type definition known in other languages where a *type* is *declared* prior to the *definition* of a variable of this type.

Variables must be defined prior to their use. These definitions must occur before any statement, thus they form the topmost part within a *statement block*.

Statements

C defines all usual flow control statements. Statements are terminated by a semicolon “;”. We can group multiple statements into blocks by enclosing them in curly brackets. Within each block, we can define new variables:

```
{  
    int i;    /* Define a global i */  
    i = 1;   /* Assign i the value 0 */  
    {      /* Begin new block */  
        int i; /* Define a local i */  
        i = 2; /* Set its value to 2 */  
    }      /* Close block */  
    /* Here i is again 1 from the outer block */  
}
```

lists all flow control statements:

Statements.

Statement	Description
break;	Leave current block. Also used to leave case statement in switch.
continue;	Only used in loops to continue with next loop immediately.
do <i>stmt</i> while (<i>expr</i>) :	Execute <i>stmt</i> as long as <i>expr</i> is TRUE.
for ([<i>expr</i>]; [<i>expr</i>]; [<i>expr</i>]	This is an abbreviation for a while loop where the first <i>expr</i> is the initialization, the second <i>expr</i> is the condition and the third <i>expr</i> is the step.
goto <i>label</i> ;	Jumps to position indicated by <i>label</i> . The destination is <i>label</i> followed by colon “:”.
if (<i>expr</i>) <i>stmt</i> [else <i>stmt</i>]	IF-THEN-ELSE in C notation.
return [<i>expr</i>];	Return from function. If function returns void return should be used with-out additional argument. Otherwise the value of <i>expr</i> is returned.
switch (<i>expr</i>) { case <i>const-expr</i> ; <i>stmts</i> case <i>const-expr</i> ; <i>stmts</i> ... [default : <i>stmts</i>] }	Ater evaluation of <i>expr</i> its value is compared with the case clauses. Execution continues at the one that matches. BEWARE : You must use break to leave the switch if you don't want execution of following case clauses! If no case clause matches and a default clause exists, the statements of the default clause are executed.
while (<i>expr</i>) <i>stmt</i>	Repeat <i>stmt</i> as long as <i>expr</i> is TRUE.

The for statement is the only statement which really differs from for statements known from other languages. All other statements more or less only differ in their syntax.

What follows are two blocks which are totally equal in their functionality. One uses the while loop while the other the for variant:

```
{
int ix, sum;
sum = 0;
ix = 0;    /* initialization */
while (ix < 10) { /* condition */
    sum = sum + 1;
    ix = ix + 1; /* step */
}
}
```

```
{
int ix, sum;
sum = 0;
for (ix = 0; ix < 10; ix = ix + 1)
    sum = sum + 1;
}
```

To understand this, you have to know, that an assignment is an expression.

Expressions and Operators

In C almost everything is an expression. For example, the assignment statement “=” returns the value of its righthand operand. As a “side effect” it also sets the value of the lefthand operand. Thus,

```
ix = 12;
```

sets the value of *ix* to 12 (assuming that *ix* has an appropriate type). Now that the assignment is also an expression, we can combine several of them; for example:

```
kx = jx = ix = 12;
```

What happens? The first assignment assigns *kx* the value of its righthand side. This is the value of the assignment to *jx*. But this is the value of the assignment to *ix*. The value of this latter is 12 which is returned to *jx* which is returned to *kx*. Thus we have expressed

```
ix = 12;  
jx = 12;  
kx = 12;
```

in one line.

Truth in C is defined as follows. The value 0 (zero) stands for FALSE. Any other value is TRUE. For example, the standard function *strcmp()* takes two strings as argument and returns -1 if the first is lower than the second, 0 if they are equal and 1 if the first is greater than the second one. To compare if two strings *str1* and *str2* are equal you often see the following if construct:

```
if (!strcmp(str1, str2)) {  
    /* str1 equals str2 */  
}  
else {  
    /* str1 does not equal str2 */  
}
```

The exclamation mark indicates the boolean NOT. Thus the expression evaluates to TRUE only if *strcmp()* returns 0.

Expressions are combined of both *terms* and *operators*. The first could be constants, variables or expressions. From the latter, C offers all operators known from other languages. However, it offers some operators which could be viewed as abbreviations to combinations of other operators. Below table lists available operators. The second column shows their priority where smaller numbers indicate higher priority and same numbers, same priority. The last column lists the order of evaluation.

Operators.

Operator	Priority	Description	Order
()	1	Function Call operator	from left
[]	1	Subscript operator	from left
->	1	Element selector	from left
!	2	Bookean NOT	from right
-	2	Binary NOT	from right
++	2	Post-/Preincrement	from right
--	2	Post-/Predecrement	from right
-	2	Unary minus	from right
(type)	2	Type cast	from right
*	2	Derefence operator	from right
&	2	Address operator	from right
sizeof	2	Size-of operator	from right
*	3	Multiplication operator	from left
/	3	Division operator	from left
%	3	Modulo operator	from left
+	4	Addition operator	from left
-	4	Subtraction operator	from left
<<	5	Left shift operator	from left
>>	5	Right shift operator	from left
<	6	Lower-than operator	from left
<=	6	Lower-or-equal operator	from left
>	6	Greater-than operator	from left
>=	6	Greater-or-equal operator	from left
==	7	Equal operator	from left
!=	7	Not-equal operator	from left
&	8	Binary AND	from left
^	9	Binary XOR	from left

Operator	Priority	Description	Order
	10	Binary OR	from left
& &	11	Boolean AND	from left
	12	Boolean OR	from left
?:	13	Conditional operator	from right
=	14	Assignment operator	from right
<i>op</i> =	14	Operator assignment operator	from right
,	15	Comma operator	from left

Most of these operators are already known to you. However, some need some more description. First of all notice that the binary boolean operators &, ^ and | are of lower priority than the equality operators == and !=. Consequently, if you want to check for bit patterns as in

```
if ((pattern & MASK) == MASK) {
    ...
}
```

you must enclose the binary operation into parenthesis.

The increment operators ++ and can be explained by the following example. If you have the following statement sequence

```
a = a + 1;
b = a;
```

you can use the preincrement operator

```
b = ++a;
```

Similarly, if you have the following order of statements:

```
b = a;
a = a + 1;
```

you can use the postincrement operator

```
b = a++;
```

Thus, the preincrement operator first increments its associated variable and then returns the new value, whereas the postincrement operator first returns the value and then increments its variable. The same rules apply to the pre- and postdecrement operator .

Function calls, nested assignments and the increment/decrement operators cause side effects when they are applied. This may introduce compiler dependencies as the evaluation order in some situations is compiler dependent. Consider the following example which demonstrates this:

```
a[i] = i++;
```

The question is, whether the old or new value of i is used as the subscript into the array a depends on the order the compiler uses to evaluate the assignment.

The conditional operator $?:$ is an abbreviation for a commonly used if statement. For example to assign max the maximum of a and b we can use the following if statement:

```
if (a > b)
    max = a;
else
    max = b;
```

These types of if statements can be shorter written as

```
max = (a > b) ? a : b;
```

The next unusual operator is the operator assignment. We are often using assignments of the following form

```
expr1 = (expr1) op (expr2)
```

for example

```
i = i * (j + 1);
```

In these assignments the lefthand value also appears on the right side. Using informal speech we could express this as “*set the value of i to the current value of i multiplied by the sum of the value of j and 1*”. Using a more natural way, we would rather say “*Multiply i with the sum of the value of j and 1*”. C allows us to abbreviate these types of assignments to

```
i *= j + 1;
```

We can do that with almost all binary operators. Note, that the above operator assignment really implements the long form although “ $j + 1$ ” is not in parenthesis.

The last unusual operator is the comma operator `,`. It is best explained by an example:

```
i = 0;
```

```
j = (i += 1, i += 2, i + 3);
```

This operator takes its arguments and evaluates them from left to right and returns the value of the rightmost expression. Thus, in the above example, the operator first evaluates “ $i += 1$ ” which, as a side effect, increments the value of i . Then the next expression “ $i += 2$ ” is evaluated which adds 2 to i leading to a value of 3. The third expression is evaluated and its value returned as the operator’s result. Thus, j is assigned 6.

The comma operator introduces a particular pitfall when using n -dimensional arrays with $n > 1$. A frequent error is to use a comma separated list of indices to try to access an element:

```
int matrix[10][5]; // 2-dim matrix
```

```
int i;
```

```
...
```

```
i = matrix[1,2]; // WON'T WORK!!
```

```
i = matrix[1][2]; // OK
```

What actually happens in the first case is, that the comma separated list is interpreted as the comma operator. Consequently, the result is 2 which leads to an assignment of the address to the third five elements of the matrix!

Some of you might wonder, what C does with values which are not used. For example in the assignment statements we have seen before,

```
ix = 12;
```

```
jx = 12;
```

```
kx = 12;
```

we have three lines which each return 12. The answer is, that C ignores values which are not used. This leads to some strange things. For example, you could write something like this:

```
ix = 1;
4711;
jx = 2;
```

But let's forget about these strange things. Let's come back to something more useful. Let's talk about *functions*.

Functions

As C is a procedural language it allows the definition of *functions*. Procedures are “simulated” by functions returning “no value”. This value is a special type called void.

Functions are declared similar to variables, but they enclose their arguments in parenthesis (even if there are no arguments, the parenthesis must be specified):

```
int sum(int to); /* Declaration of sum with one argument */
int bar();      /* Declaration of bar with no arguments */
void foo(int ix, int jx);
                /* Declaration of foo with two arguments */
```

To actually define a function, just add its body:

```
int sum(int to) {
    int ix, ret;
    ret = 0;
    for (ix = 0; ix < to; ix = ix + 1)
        ret = ret + ix;
    return ret; /* return function's value */
} /* sum */
```

C only allows you to pass function arguments by value. Consequently you cannot change the value of one argument in the function. If you must pass an argument by reference you must program it on your own. You therefore use *pointers*.

Pointers and Arrays

One of the most common problems in programming in C (and sometimes C++) is the understanding of pointers and arrays. In C (C++) both are highly related with some small but essential differences. You declare a pointer by putting an asterisk between the data type and the name of the variable or function:

```
char *strp;    /* strp is 'pointer to char' */
```

You access the content of a pointer by dereferencing it using again the asterisk:

```
*strp = 'a';    /* A single character */
```

As in other languages, you must provide some space for the value to which the pointer points. A pointer to characters can be used to point to a sequence of characters: the *string*. Strings in C are terminated by a special character NUL (0 or as char ‘\0’). Thus, you can have strings of any length. Strings are enclosed in double quotes:

```
strp = "hello";
```

In this case, the compiler automatically adds the terminating NUL character. Now, *strp* points to a sequence of 6 characters. The first character is ‘h’, the second ‘e’ and so forth. We can access these characters by an index in *strp*:

```
strp[0] /* h */
```

```
strp[1] /* e */
```

```
strp[2] /* l */
```

```
strp[3] /* l */
```

```
strp[4] /* o */
```

```
strp[5] /* \0 */
```

The first character also equals “**strp*” which can be written as “**(strp + 0)*”. This leads to something called *pointer arithmetic* and which is one of the powerful features of C. Thus, we have the following equations:

```
*strp == *(strp + 0) == strp[0]
```

```
*(strp + 1) == strp[1]
```

```
*(strp + 2) == strp[2]
```

```
...
```

Note that these equations are true for any data type. The addition is **not** oriented to bytes, it is oriented to the size of the corresponding pointer type!

The *strp* pointer can be set to other locations. Its destination may *vary*. In contrast to that, *arrays* are *fix* pointers. They point to a predefined area of memory which is specified in brackets:

```
char str[6];
```

You can view *str* to be a constant pointer pointing to an area of 6 characters. We are **not** allowed to use it like this:

```
str = "hallo"; /* ERROR */
```

because this would mean, to change the pointer to point to 'h'. We must copy the string into the provided memory area. We therefore use a function called strcpy() which is part of the standard C library.

```
strcpy(str, "hallo"); /* Ok */
```

Note however, that we can use *str* in any case where a pointer to a character is expected, because it is a (fixed) pointer.

A First Program

Here we introduce the first program which is so often used: a program which prints "Hello, world!" to your screen:

```
#include <stdio.h>
```

```
/* Global variables should be here */
```

```
/* Function definitions should be here */
```

```
int
```

```
main() {
```

```
    puts("Hello, world!");
```

```
    return 0;
```

```
} /* main */
```

The first line looks something strange. Its explanation requires some information about how C (and C++) programs are handled by the compiler. The compilation step is roughly divided into two steps. The first step is called “preprocessing” and is used to prepare raw C code. In this case this step takes the first line as an argument to include a file called *stdio.h* into the source. The angle brackets just indicate, that the file is to be searched in the standard search path configured for your compiler. The file itself provides some declarations and definitions for standard input/output. For example, it declares a function called *put()*. The preprocessing step also deletes the comments.

In the second step the generated raw C code is compiled to an executable. Each executable must define a function called *main()*. It is this function which is called once the program is started. This function returns an integer which is returned as the program’s exit status.

Function *main()* can take arguments which represent the command line parameters. We just introduce them here but do not explain them any further:

```
#include <stdio.h>

int
main(int argc, char *argv[]) {
    int ix;
    for (ix = 0; ix < argc; ix++)
        printf(“My %d. argument is %s\n”, ix, argv[ix]);
    return 0;
} /* main */
```

The first argument *argc* just returns the number of arguments given on the command line. The second argument *argv* is an array of strings. (Recall that strings are represented by pointers to characters. Thus, *argv* is an array of pointers to characters.)

What Next?

This section is far from complete. We only want to give you an expression of what C is. We also want to introduce some basic concepts which we will use in the following

section. Some concepts of C are improved in C++. For example, C++ introduces the concept of *references* which allow something similar to call by reference in function calls.

We suggest that you take your local compiler and start writing a few programs (if you are not already familiar with C, of course). One problem for beginners often is that existing library functions are unknown. If you have a UNIX system try to use the man command to get some descriptions. Especially you might want to try:

man gets

man printf

man puts

man scanf

man strcpy

We also suggest, that you get yourself a good book about C (or to find one of the on-line tutorials). We try to explain everything we introduce in the next sections. However, there is nothing wrong with having some reference at hand.

8 From C To C++

Peter Müller Globewide Network Academy (GNA) pmueller@uu-gna.mit.edu

This section presents extensions to the C language which were introduced by C++ [6]. It also deals with object-oriented concepts and their realization.

Basic Extensions

The following sections present extensions to already introduced concepts of C.

C++ adds a new comment which is introduced by two slashes (//) and which lasts until the end of line. You can use both comment styles, for example to comment out large blocks of code:

```
/* C comment can include // and can span over
```

```
several lines. */
```

```
/** This is the C++ style comment */ until end of line
```

In C you must define variables at the beginning of a block. C++ allows you to define variables and objects at any position in a block. Thus, variables and objects should be defined where they are used.

Data Types

C++ introduces a new data type called *reference*. You can think of them as if they were “aliases” to “real” variables or objects. As an alias cannot exist without its corresponding real part, you cannot define single references. The ampersand (&) is used to define a reference. For example:

```
int ix;    /* ix is “real” variable */  
  
int &rx = ix; /* rx is “alias” for ix */  
  
ix = 1;    /* also rx == 1 */  
rx = 2;    /* also ix == 2 */
```

References can be used as function arguments and return values. This allows to pass parameters as reference or to return a “handle” to a calculated variable or object.

The provides you with an overview of possible declarations. It is not complete in that it shows not every possible combination and some of them have not been introduced here, because we are not going to use them. However, these are the ones which you will probably use very often.

Declaration expressions.

Declaration	<i>name</i> is ...	Example
<i>type</i> name ;	<i>type</i>	int count ;
<i>type</i> name [] ;	(open) array of <i>type</i>	int count [] ;
<i>type</i> name [<i>n</i>] ;	array with <i>n</i> elements of type <i>type</i> (name[0], name [1], ..., name [n-1])	int count [3] ;
<i>type</i> * name ;	pointer to <i>type</i>	int * count ;
<i>type</i> * name [] ;	(open) array of pointers to <i>type</i>	int * count [] ;
<i>type</i> * (name []) ;	(open) array of pointers to <i>type</i>	int * (count []) ;
<i>type</i> (* name) [] ;	pointer to (open) array of <i>type</i>	int (* count) [] ;
<i>type</i> & name ;	reference to <i>type</i>	int & count ;
<i>type</i> name () ;	function returning <i>type</i>	int count () ;
<i>type</i> * name () ;	function retruning pointer to <i>type</i>	int * count () ;

Declaration	<i>name is ...</i>	Example
<i>type</i> * (name ()) ;	function returning pointer to <i>type</i>	int * (count ());
<i>type</i> (* name) () ;	pointer to function returning <i>type</i>	int (*count) () ;
<i>type</i> & name ();	function returning reference to <i>type</i>	int & count () ;

In C and C++ you can use the modifier `const` to declare particular aspects of a variable (or object) to be constant. The next table 8.2 lists possible combinations and describe their meaning. Subsequently, some examples are presented which demonstrate the use of `const`.

Constant declaration expressions.

Declaration	<i>name is</i>
<code>const <i>type</i> name = <i>value</i> ;</code>	constant <i>type</i>
<code><i>type</i> * const name = <i>value</i> ;</code>	constant pointer to <i>type</i>
<code>const <i>type</i> * name = <i>value</i> ;</code>	(variable) pointer to constant <i>type</i>
<code>const <i>type</i> * const name = <i>value</i> ;</code>	constant pointer to constant <i>type</i>

Now let's investigate some examples of constant variables and how to use them. Consider the following declarations (again from [1]):

```
int i;           // just an ordinary integer
int *ip;        // uninitialized pointer to
                // integer
int * const cp = &i;    // constant pointer to integer
const int ci = 7;     // constant integer
const int *cip;       // pointer to constant integer
const int * const cicp = &ci; // constant pointer to constant
                        // integer
```

The following assignments are **valid**:

```
i = ci;    // assign constant integer to integer
*cp = ci;  // assign constant integer to variable
           // which is referenced by constant pointer
```

```

cip = &ci; // change pointer to constant integer
cip = cicp; // set pointer to constant integer to
           // reference variable of constant pointer to
           // constant integer

```

The following assignments are **invalid**:

```

ci = 8; // cannot change constant integer value
*cip = 7; // cannot change constant integer referenced
         // by pointer
cp = &ci; // cannot change value of constant pointer
ip = cip; // this would allow to change value of
         // constant integer *cip with *ip

```

When used with references some peculiarities must be considered. See the following example program:

```

#include <stdio.h>

int main() {
    const int ci = 1;
    const int &cr = ci;
    int &r = ci; // create temporary integer for reference
    // cr = 7; // cannot assign value to constant reference
    r = 3; // change value of temporary integer
    print("ci == %d, r == %d\n", ci, r);
    return 0;
}

```

When compiled with GNU g++, the compiler issues the following warning:
conversion from 'const int' to 'int &' discards const

What actually happens is, that the compiler automatically creates a temporary integer variable with value of *ci* to which reference *r* is initialized. Consequently, when changing *r* the value of the temporary integer is changed. This temporary variable lives as long as reference *r*.

Reference *cr* is defined as *read-only* (constant reference). This disables its use on the left side of assignments. You may want to remove the comment in front of the particular line to check out the resulting error message of your compiler.

Functions

C++ allows function overloading as defined in Polymorphism. For example, we can define two different functions *max()*, one which returns the maximum of two integers and one which returns the maximum of two strings:

```
#include <stdio.h>

int max(int a, int b) {
    if (a > b) return a;
    return b;
}

char *max(char *a, char * b) {
    if (strcmp(a, b) > 0) return a;
    return b;
}

int main() {
    printf("max(19, 69) = %d\n", max(19, 69));
    printf("max(abc, def) = %s\n", max("abc", "def"));
    return 0;
}
```


The above example program defines these two functions which differ in their parameter list, hence, they define two different functions. The first *printf()* call in function *main()* issues a call to the first version of *max()*, because it takes two integers as its argument. Similarly, the second *printf()* call leads to a call of the second version of *max()*.

References can be used to provide a function with an alias of an actual function call argument. This enables to change the value of the function call argument as it is known from other languages with call-by-reference parameters:

```
void foo(int byValue, int &byReference) {  
    byValue = 42;  
    byReference = 42;  
}
```

```
void bar() {  
    int ix, jx;  
  
    ix = jx = 1;  
    foo(ix, jx);  
    /* ix == 1, jx == 42 */  
}
```

First Object-oriented Extensions

In this section we present how the object-oriented concepts used in C++.

Classes and Objects

C++ allows the declaration and definition of classes. Instances of classes are called *objects*. Recall the drawing program example of “a - kind - of” relationship. There we have developed a class *Point*. In C++ this would look like this:

```
class Point {  
    int _x, _y;    // point coordinates
```

```

public:          // begin interface section
    void setX(const int val);
    void setY(const int val);
    int getX() { return _x; }
    int getY() { return _y; }
};

```

Point apoint

This declares a class *Point* and defines an object *apoint*. You can think of a class definition as a structure definition with functions (or “methods”). Additionally, you can specify the *access rights* in more detail. For example, *_x* and *_y* are **private**, because elements of classes are private as default. Consequently, we explicitly must “switch” the access rights to declare the following to be **public**. We do that by using the keyword **public** followed by a colon: Every element following this keyword are now accessible from outside of the class.

We can switch back to private access rights by starting a private section with **private**:. This is possible as often as needed:

```

class Foo {
    // private as default ...

public:
    // what follows is public until ...

private:
    // ... here, where we switch back to private ...

public:
    // ... and back to public.

```

```
};
```

Recall that a structure `struct` is a combination of various data elements which are accessible from the outside. We are now able to express a structure with help of a class, where all elements are declared to be public:

```
class Struct {  
public:    // Structure elements are public by default  
    // elements, methods  
};
```

This is exactly what C++ does with `struct`. Structures are handled like classes. Whereas elements of classes (defined with `class`) are private by default, elements of structures (defined with `struct`) are public. However, we can also use `private:` to switch to a private section in structures.

Let's come back to our class *Point*. Its interface starts with the public section where we define four methods. Two for each coordinate to set and get its value. The set methods are only declared. Their actual functionality is still to be defined. The get methods have a function body: They are defined *within* the class or, in other words, they are *inlined methods*.

This type of method definition is useful for small and simple bodies. It also improve performance, because bodies of inlined methods are “copied” into the code wherever a call to such a method takes place.

On the contrary, calls to the set methods would result in a “real” function call. We define these methods outside of the class declaration. This makes it necessary, to indicate to which class a method definition belongs to. For example, another class might just define a method *setX()* which is quite different from that of *Point*. We must be able to define the *scope* of the definition; we therefore use the scope operator “`::`”:

```
void Point::setX(const int val) {  
    _x = val;  
}
```

```
void Point::setY(const int val) {
```

```
_y = val;  
}
```

Here we define method *setX()* (*setY()*) within the scope of class *Point*. The object *apoint* can use these methods to set and get information about itself:

Point apoint;

```
apoint.setX(1); // Initialization  
apoint.setY(1);  
  
//  
// x is needed from here, hence, we define it here and  
// initialize it to the x-coordinate of apoint  
//  
  
int x = apoint.getX();
```

The question arises about how the methods “know” from which object they are invoked. This is done by implicitly passing a pointer to the invoking object to the method. We can access this pointer within the methods as this. The definitions of methods *setX()* and *setY()* make use of class members *_x* and *_y*, respectively. If invoked by an object, these members are “automatically” mapped to the correct object. We could use this to illustrate what actually happens:

```
void Point::setX(const int val) {  
    this->_x = val; // Use this to reference invoking  
                // object  
}  
  
void Point::setY(const int val) {
```

```

this->_y = val;
}

```

Here we explicitly use the pointer `this` to explicitly dereference the invoking object. Fortunately, the compiler automatically “inserts” these dereferences for class members, hence, we really can use the first definitions of `setX()` and `setY()`. However, it sometimes make sense to know that there is a pointer `this` available which indicates the invoking object.

Currently, we need to call the set methods to initialize a point object. However, we would like to initialize the point when we define it. We therefore use special methods called *constructors*.

How to Write a Program

Until now, we have only presented parts of or very small programs which could easily be handled in one file. However, greater projects, say, a calendar program, should be split into manageable pieces, often called *modules*. Modules are implemented in separate files and we will now briefly discuss how modularization is done in C and C++. This discussion is based on UNIX and the GNU C++ compiler. If you are using other constellations the following might vary on your side. This is especially important for those who are using integrated development environments (IDEs), for example, Borland C++.

Roughly speaking, modules consist of two file types: *interface descriptions* and *implementation files*. To distinguish these types, a set of suffixes are used when compiling C and C++ programs. The below shows some of them.

Extensions and file types.

Extension (s)	File Type
.h, .hxx, .hpp	interface descriptions (“header” or “include files”)
.c	implementation files of C
.cc, .c, .cxx, .cpp .c++	implementation files of C++
.tpl	interface description (templates)

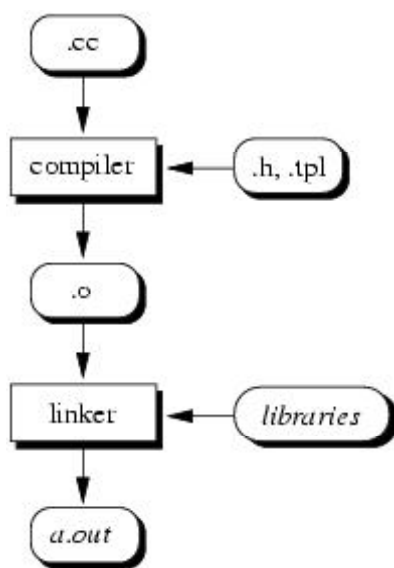
In this tutorial we will use `.h` for header files, `.cc` for C++ files and `.tpl` for template definition files. Even if we are writing “only” C code, it makes sense to use `.cc` to force the compiler to treat it as C++. This simplifies combination of both, since the internal mechanism of how the compiler arrange names in the program differs between both languages.

Compilation Steps

The compilation process takes `.cc` files, preprocess them (removing comments, add header files) and translates them into *object files*. Typical suffixes for that file type are `.o` or `.obj`.

After successful compilation the set of object files is processed by a *linker*. This program combine the files, add necessary libraries and creates an executable. Under UNIX this file is called *a.out* if not other specified. These steps are illustrated in below figure.

Compilation steps.



With modern compilers both steps can be combined. For example, our small example programs can be compiled and linked with the GNU C++ compiler as follows (“example.cc” is just an example name, of course):

```
gcc example.cc
```

A Note about Style

Header files are used to describe the interface of implementation files. Consequently, they are included in each implementation file which uses the interface of the particular implementation file. As mentioned in previous sections this inclusion is achieved by a copy of the content of the header file at each preprocessor `#include` statement, leading to a “huge” raw C++ file.

To avoid the inclusion of multiple copies caused by mutual dependencies we use *conditional coding*. The preprocessor also defines conditional statements to check for various aspects of its processing. For example, we can check if a macro is already defined:

```
#ifndef MACRO  
  
#define MACRO /* define MACRO */  
  
...  
  
#endif
```

The lines between `#ifndef` and `#endif` are only included, if `MACRO` is not already defined. We can use this mechanism to prevent multiple copies:

```
/*  
** Example for a header file which 'checks' if it is  
** already included. Assume, the name of the header file  
** is 'myheader.h'  
*/
```

```
#ifndef __MYHEADER_H  
#define __MYHEADER_H
```

```
/*  
** Interface declarations go here  
*/
```

```
#endif /* __MYHEADER_H */
```

`__MYHEADER_H` is a unique name for each header file. You might want to follow the convention of using the name of the file prefixed with two underbars. The first time the file is included, `__MYHEADER_H` is not defined, thus every line is included and processed. The first line just defines a macro called `__MYHEADER_H`. If accidentally the file should be included a second time (while processing the same input file), `__MYHEADER_H` is defined, thus everything leading up to the `#endif` is skipped.

Exercises

1.

Polymorphism. Explain why

```
void display(const DrawableObject obj);
```

does not produce the desired output.

The List - A Case Study

Peter Müller

Globewide Network Academy (GNA)

pmueller@uu-gna.mit.edu

Generic Types (Templates)

In C++ generic data types are called *class templates* or just *templates* for short. A class template looks like a normal class definition, where some aspects are represented by *placeholders*. In the forthcoming list example we use this mechanism to generate lists for various data types:

```
template <class T>
class List : ... {
public:
    ...
    void append(const T data);
    ...
};
```

In the first line we introduce the keyword `template` which starts every template declaration. The arguments of a template are enclosed in angle brackets.

Each argument specifies a placeholder in the following class definition. In our example, we want class *List* to be defined for various data types. One could say, that we want to define a *class of lists*. In this case the class of lists is defined by the type of objects they contain. We use the name *T* for the placeholder. We now use *T* at any place where normally the type of the actual objects are expected. For example, each list provides a method to append an element to it. We can now define this method as shown above with use of *T*.

An actual list *definition* must now specify the type of the list. If we stick to the class expression used before, we have to *create a class instance*. From this class instance we can then create “real” object instances:


```
List<int> integerList;
```

Here we create a class instance of a *List* which takes integers as its data elements. We specify the type enclosed in angle brackets. The compiler now applies the provided argument “int” and automatically generates a class definition where the placeholder *T* is replaced by *int*, for example, it generates the following method declaration for *append()*:

```
void append(const int data);
```

Templates can take more than one argument to provide more placeholders. For example, to declare a dictionary class which provides access to its data elements by a key, one can think of the following declaration:

```
template <class K, class T>
class Dictionary {
    ...
public:
    ...
    K getKey(const T from);
    T getData(const K key);
    ...
};
```

Here we use two placeholders to be able to use dictionaries for various key and data types.

Template arguments can also be used to generate parameterized class definitions. For example, a stack might be implemented by an array of data elements. The size of the array could be specified dynamically:

```
template <class T, int size>
```

```
class Stack {
```

```
    T _store[size];
```

```
public:
```

```
    ...
```

```
};
```

```
Stack<int,128> mystack;
```

In this example, *mystack* is a stack of integers using an array of 128 elements. However, in the following we will not use parameterized classes.
