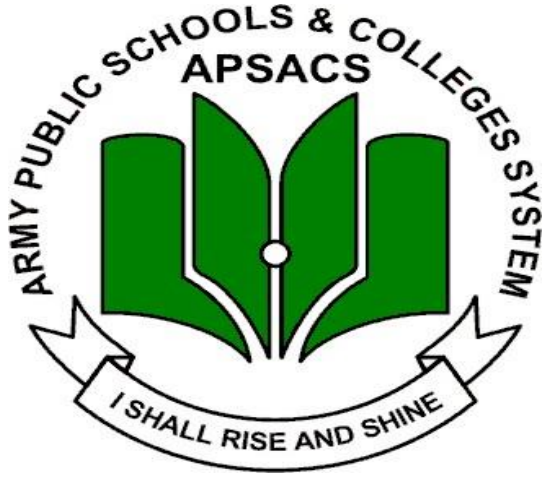


APS & C For Girls (Azam Garrison) Computer Science Notes XII-HSSC

OPERATING SYSTEM

DEFINITION:

"A software on the hard drive that enables the computer hardware to communicate with the system software and operate it is called the operating system"

TYPES OF OPERATING SYSTEMS

The following are the important types of operating systems that are commonly used on various computer systems.

BATCH PROCESSING OPERATING SYSTEM

A batch processing operating system is a software that groups together the same type of jobs in batches and automatically executes them one by one. It performs the same type of task on all the jobs in a batch in the sequence in which they appear. It provides an easy and efficient way of processing the same type of jobs. For example, at the end of the month, banks print statements for each account holder. A batch processing system can easily and efficiently print each account holder's statement one by one.

MULTIPROGRAMMING OPERATING SYSTEM

A multiprogramming operating system is a software that loads one or more programs in main memory and executes them using a single CPU (Central Processing Unit). The CPU executes only one program at a time while other programs are waiting in the queue. In a multiprogramming system when one program is busy with input/output operation, the CPU executes another program that is in the queue. In this way, the multiprogramming operating system uses the CPU time and other resources of the computer to improve the performance of the computer.

MULTITASKING OPERATING SYSTEM

A multitasking operating system is a software that performs multiple tasks at the same time on a computer that has a single CPU. The CPU executes only one program at a time but it rapidly switches between multiple programs and it appears as if all the users' programs are being executed at the same time.

TIME-SHARING OPERATING SYSTEM

A time-sharing operating system is a software that shares the CPU time between multiple programs that are loaded in main memory. A time-sharing operating system gives a very short period of CPU time to each program one by one. This short period is called time slice or quantum. Since the CPU is switched between the programs at an extremely fast speed, all the users get the impression of having their CPU. It is used in mini and mainframe computers that support a large number of users in a big organization such as airline, bank, university, etc.

REAL-TIME OPERATING SYSTEM

A real-time operating system is a software that runs real-time applications that must process data as soon as it comes and provides an immediate response. The real-time operating system executes special applications within the specified time with high reliability. It is commonly used in space research programs, real-time traffic control and to control industrial processes such as oil refining.

MULTIPROCESSOR OPERATING SYSTEM

A multiprocessor operating system is a software that controls the operations of two or more CPUs within a single computer system. All the CPUs of the computer share the same main memory and input/output devices. Multiprocessing operating systems are used to obtain a very high speed to process a large amount of data. It executes a single program using many CPUs at the same time to improve processing speed. Computers that support multiprocessing have a sophisticated architecture which is difficult to design.

PARALLEL PROCESSING OPERATING SYSTEM

A parallel processing operating system is a software that executes programs developed in a parallel programming language. It uses many processors at the same time. In a parallel processing system, the task of a program that requires many calculations is divided into many smaller tasks and these are processed by multiple processors at the same time. Parallel processing operating systems are used in supercomputers that have thousands of processors.

DISTRIBUTED OPERATING SYSTEM

A distributed operating system is a software that manages the operation of a distributed system. A distributed system allows the execution of application software on different computers in a network. In a distributed system, user programs may run on any computer in the network and access data on any other computer. The users of the distributed system do not know on which computer their programs are running. The distributed operating system automatically balances the load on different computers in the network and provides fast execution of application software.

EMBEDDED OPERATING SYSTEM

An embedded operating system is a built-in operating system which is embedded in the hardware of the device. It controls the operation of devices such as microwave ovens, TV, camera, washing machine, games, etc. It runs automatically when the device is turned on and performs a specific task.

---

a) Single user and multi-user operating system

b) Threads and process

c) Multiprogramming and Multiprocessing

Answer:

a) DIFFERENCE BETWEEN SINGLE-USER AND MULTI-USER OPERATING SYSTEMS

Operating systems are divided into single-user and multi-user operating systems based on the number of users, they can support.

SINGLE-USER OPERATING SYSTEM

The operating system that allows only one person to operate the computer at a time is known is a single-user operating system. Commonly used single-user operating systems are DOS and Windows.

MULTI-USER OPERATING SYSTEM

The operating system that allows many users on different terminals or microcomputers to use the resources of a single central computer (server) in a network is known as a multi-user operating

system. It is used on servers in business and offices where many users have to access the same application software and other resources. Some examples of multi-user operating systems are UNIX, Linux, Windows 2000 onward and Mac OS X**.**

b) THREADS AND PROCESS

The differences are as follows:

1. Processes are independent of one another while threads are not independent of one another
2. Unlike processes, all threads can access every address in the task
3. Processes might or might not assist one another because processes may originate from different users while threads are designed to assist one another

c) MULTIPROGRAMMING AND MULTIPROCESSING

MULTIPROGRAMMING

In multiprogramming, many programs are loaded in memory but the CPU only executes one program at a time. Other programs wait until the previous program is executed out or blocked.

For example, when a user loads program 1 (say MS-Word) and program 2 (say C- language compiler). The CPU can execute only one program i.e., MS-Word or C-language compiler.

The advantage of multiprogramming is that it saves the user's time in loading the programs to the main memory and runs the programs quickly. The only drawback is, the system requires more main memory as it is occupied by many programs. Sometimes bigger programs cannot fully load in main memory and thus programs run slowly.

MULTIPROCESSING

Multiprogramming is a rudimentary form of parallel processing in which several programs are run at the same time on a uniprocessor.

Multiprogramming is used for Multiuser Systems (Servers)

If we are running multiple processes on a Multi-User system like a Server to which multiple computers are connected then it is known as Multiprogramming.

DISTRIBUTED OPERATING SYSTEM

A distributed operating system is a software that manages the operation of a distributed system. A distributed system allows the execution of application software on different computers in a network. In a distributed system, user programs may run on any computer in the network and access data on any other computer. The users of the distributed system do not know on which computer their programs are running. The distributed operating system automatically balances the load on different computers in the network and provides fast execution of application software.

ADVANTAGES:

The advantages of a distributed operating system are as follows:

- Communication and resource sharing is possible which eliminates the need for dedicated resources with each computer system.

- The distributed systems are economical in the sense that expensive resources are shared.
- The system is reliable because of the availability of multiple machines for performing the task of a failed system.
- The system has the potential for incremental growth

DISADVANTAGES:

The disadvantages of a distributed operating system are as follows:

- Network connectivity is an essential part of a distributed system which is a difficult and expensive task.
- Security and privacy is an issue.

---

a) STATES OF A PROCESS

There are five states of a process which are new, ready, running, waiting and terminated

1. **New State**
2. A process is in a terminated state when it completes its execution. This is the first state of a process when it is created. Any new operation or service that is requested by a program for execution by the processor is known as a new state of the process.
3. **Ready State**
4. A process is said to be in the ready state when it is ready for execution but it is waiting to be assigned to the processor by the operating system.
5. **Running State**
6. A process is said to be in running state when it is being executed by the processor. A process is assigned to a processor for execution by the operating system.
7. **Blocked State/ Waiting State**
8. A process is in a blocked or waiting state when it is not under execution. It is waiting for a resource to become available.
9. **Terminated State**
10. A process is in the terminated state when it completes its execution

b) DOS

DOS

ABBREVIATION:

It is an abbreviation of Disk Operating System.

HISTORY:

It is a single-user operating system and has been very popular on microcomputers up to mid-1990s. DOS was designed by IBM (International Business Machines). DOS resides on disk and controls the overall functioning of the computer.

FUNCTION:

It performs the following major tasks:

- Control input and output devices
- Execute user programs
- Manage system resources

- Provide user interface
- Memory management

c) REAL-TIME OPERATING SYSTEM

A real-time operating system is a software that runs real-time applications that must process data as soon as it comes and provides an immediate response. The real-time operating system executes special applications within the specified time with high reliability. It is commonly used in space research programs, real-time traffic control and to control industrial processes such as oil refining.

d) EMBEDDED OPERATING SYSTEM

An embedded operating system is a built-in operating system which is embedded in the hardware of the device. It controls the operation of devices such as microwave oven, TV, camera, washing machine, games, etc. It runs automatically when the device is turned on and performs a specific task.

FUNCTIONS OF OPERATING SYSTEM

The following are the main functions of the operating system.

- – Process Management
- – Memory Management
- – Input/Output Management
- – File Management
- – Resource Management
- – User Management

Process Management

Process management is an essential part of the operating system (OS). A process is a program in execution. In a computer system, multiple processes are executing concurrently or waiting for their turn to be executed. A process in execution needs resources like processing resources, memory, and I/O resources. The OS must allocate resources to processes, enable processes to share and exchange information, and protect the resources of each process from other processes.

Memory Management

Memory management is the process of allocating memory space for user programs in main memory. When programs are run by users, the operating system allocates portions of free memory to programs. When a program is closed, the operating system will free the memory portion used by that program for reuse. The operating system automatically loads user programs in available memory space and executes them.

Input/Output Management

Input/output management is the process of controlling the operation of all the input/output devices attached to the computer. The user communicates with a computer through various input/output devices such as a keyboard, mouse, monitor printer, etc. Management of these devices is the responsibility of the operating system. The operating system uses the Input/Output controller to manage and coordinate the operation of all the input/output devices.

File Management

A file management system is part of an operating system that organizes stores and keeps track of the computer files and folders. Computer files can be documents, programs, images, videos, etc. The operating system controls the common operations performed on files. These operations include creating, opening, editing, renaming, moving, copying, deleting and searching files.

Resource Management

The operating system automatically manages the resources of a computer when application programs are executed by the computer user. The resources of a computer include microprocessors, memory and all the devices attached to the computer. The operating system allocates resources of a computer to the application program according to the user's requirement in an efficient way to improve the performance of the computer.

User Management

User management is an important feature of an operating system for maintaining a secure computer system. The operating system gives full control over a computer system to a person known as the administrator. An administrator installs various programs on the computer system for users. He also creates and manages user accounts. When a user account is created, the user is assigned a user name and a password. Administrator allows the users to run various application programs that are installed on the computer. A user can log in to the computer system by entering the username and password, run programs and save his files in his personal folder. The operating system does not allow users to install programs or create new users.

DOS, Windows and UNIX operating systems

DOS

ABBREVIATION:

It is an abbreviation of Disk Operating System.

HISTORY:

It is a single-user operating system and has been very popular on microcomputers up to the mid-1990s. DOS was designed by IBM (International Business Machines). DOS resides on disk and controls the overall functioning of the computer.

WINDOWS OPERATING SYSTEM

Windows is the most popular operating system used on microcomputers. It was developed by Microsoft. Many different versions of Windows Operating systems were developed and used successfully in the past. Some of these versions are Windows 95, XP etc. The latest version is Windows 10.

UNIX

UNIX is a multi-user CLI operating system introduced in 1969. It allows multiple users to run different programs at the same time. UNIX was developed for use on the large computer system(Mainframe). It uses a command-line interface but later Graphical User Interface was also introduced.

Chapter 2: System Development Life Cycle

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC):

DEFINITION:

The software development life cycle (SDLC) is a conceptual model used in project management that describes the stages involved in an information system development project from an initial feasibility study through maintenance of the completed application.

OBJECTIVES OF SDLC:

The objectives of the software Development Life Cycle (SDLC) are as follows:

1. Delivery of quality software that meets customer expectations.
2. Delivery of inexpensive and cost-effective software which are easily maintainable.
3. Maximize productivity in terms of the software systems delivered.
4. One of the major objectives of SDLC is to establish an appropriate level of management authority to direct, coordinate, control, review and approve the software development project.
5. SDLC should ensure project management accountability.
6. Proper documentation of all the requirements needed for the development of a new software system.
7. Ensuring that projects are developed within the current and planned information technology infrastructure.
8. SDLC should identify the potential project risks in advance so that the proper planning should be done.

SYSTEM: EXPLANATION:

The term "system" is originated from the Greek term system, which means to "place together." It can be defined as a set of interrelated components having a clearly defined boundary that works together to achieve a common set of objectives.

A system can be developed by applying a set of methods, procedures and routine in a proper sequence to carry out some specific tasks. When all these functions are applied to build software then the system will be called a software system

BEGINNING OF TESTING ACTIVITY:

The testing activity starts from the initial stage i.e. From requirement analysis.

Importance of Software Development Life Cycle

Following points summarize the importance of the use of SDLC:

- SDLC is important because it breaks down the entire life cycle of software development into phases thus making it easier for the development team members to easily evaluate each part of software development.
- SDLC makes it easier for programmers to work concurrently on each phase.
- It provides a rough time estimate that when the software will be available for use.
- It delivers quality software which meets or exceeds customer expectations.
- It provides the basic framework for the development of quality software.
- SDLC helps the project managers to establish project management and be followed strictly during system development.

- SDLC clearly defines and assigns the roles and responsibilities of all the involved parties.
- It ensures that the requirements for the development of the software system are well defined and subsequently satisfied

STAKEHOLDERS OF SDLC:

Those entities which are either within the organization or outside of the organization that sponsors a project, or have an interest or have the intention to get it after its successful completion, or may have a positive or negative influence in the project completion called **stakeholders.** Project stakeholders include the customer, the user group, the project manager, the development team and the testers.

All those who have some interest in the project can be considered as stakeholders of that project. The individuals as well as the organizations that are actively involved in the project, or whose interest may be affected as a result of project execution or project completion are the part of stakeholders. The project management team must identify the stakeholders determine their requirements, expectations and mange their influence about the requirements to ensure a successful project.

RESPONSIBILITIES OF STAKEHOLDERS:

The basic role of the stakeholders are:

- – For the development of software, resources such as time, money, equipment etc are needed which should be provided to the project team by the stakeholders.
- – Stakeholders educate the developers about their business.
- – They spend more time to provide information and clarify requirements to the analyst and developers.
- – The stakeholders should be specific and precise about the requirements.
- – Make timely decisions.
- – Respect a developer's assessment of cost and feasibility.
- – Set requirements priorities.
- – Review and provide timely feedback.
- Promptly communicate changes to requirements.

SOFTWARE REQUIREMENTS:

Software requirements is a field within software engineering that deals with establishing the needs of stakeholders that are to be solved by software.

DIFFERENCE BETWEEN FUNCTIONAL AND NON FUNCTIONAL REQUIREMENTS:

FUNCTIONAL REQUIREMENTS:

Functional requirements are those requirements of a software system which describes a function of a software system or its component. It includes calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Examples of functional requirements are:

a) Interface requirements

- – Field accepts numeric data entry

- – Field only accepts dates before the current date.
- – Screen can print on-screen data to the printer

b) Business requirements:

- – Data must be entered before a request can be approved.

c) Regulatory/Compliance requirements:

- – The database will have a functional audit trail.
- – The system will limit access to authorized users.
- – The spreadsheet can secure data with an electronic signature.

d) Security requirements:

- – Member of the data entry group can enter requests but not approve or delete requests.
- – Members of the /managers group can enter or approve a request, but not delete requests.
- – Members of the administrator group cannot enter or approve a request but can delete request.

NON FUNCTIONAL REQUIREMENTS:

Non- functional requirements are those requirements which specify criteria for the judgement of the operations of a system. It describes how well the system performs its duties. Non-functional requirements are often called qualities of a system. These requirements depend upon the nature of the software.

Different types of non-functional requirements are:

- 1. Accessibility Requirements
- 2. Accuracy Requirements
- 3. Backup and Recovery Requirements
- 4. Memory capacity Requirements
- 5. Compatibility Requirements
- 6. Error- handling Requirements
- 7. Maintainability Requirements
- 8. Performance Requirements
- . Security Requirements

Chapter 3: Object Oriented Programming using C++

Pick that variable from the following list which is improperly declared and named. Also, describe the reason of its invalidity.

- **1) int sum;**
- **2) int cats=5, dogs=5;**
- **3) int my variable name;**
- **4) int void= 5;**
- **6) int 3some;**
- **7) int meters_of_pipe;**
- **8) int length , width=5;**

Answer:

Following are the variables which are improperly declared and named.

- **int my variable name;**
  Reason:
  It is having the spaces between the names.
- **int void= 5;**
  Reason:
  Reserved or keywords cannot be used as the variable names.
- **int 3some;**
  Reason:
  A variable name starting with an integer is not valid.

Write a C++ program to get six subject marks of a student and then calculate its total, average, and percentage and display them on screen.

Answer:

```
#include <stdio.h>
void main() {
   int subj1, subj2, subj3, subj4, subj5, subj6;
   float total, average, percentage;
   cout << "Enter marks of the following Subjects (Max: 200)" << endl;
   cout << "————————————————————————————————" << endl;
```

```
    cout << "Subject # 01" << endl;
    cin >> subj1;
    cout << "Subject # 02" << endl;
    cin >> subj2;
    cout << "Subject # 03" << endl;
    cin >> subj3;
    cout << "Subject # 04" << endl;
    cin >> subj4;
    cout << "Subject # 05" << endl;
    cin >> subj5;
    cout << "Subject # 06" << endl;
    cin >> subj6;
    total = subj1 + subj2 + subj3 + subj4 + subj5 + subj6;
    average = total / 6;
    percentage = 100 * (total / 1200);
    cout << "The total marks of the student = " << total << endl;
    cout << "The Average marks of the student = " << average << endl;
    cout << "The Percentage of the student = " << percentage << endl;
}
```

Write a C++ program to find out maximum value out of three integers using conditional operator.

Answer:

```
void main() {
    int number1, number2, number3;
    cout << "PROGRAM TO PRINT MAXIMUM OF THREE NUMBERS" << endl;
    cout << "————————————————————————————" << endl;
    cout << "Enter First Number : " << endl;'
    cin >> number1;
    cout << "Enter Second Number : " << endl;
    cin >> number2;
    cout << "Enter Third Number : " << endl;
    cin >> number3;
    if (number1 > number2 && number1 > number3)
    {
        cout << "The maximum number is: " << number1 << endl;
    }
    if (number2 > number1 && number2 > number3)
```

```
    {
        cout << "The maximum number is: " << number2 << endl;
    }
    if (number3 > number1 && number3 > number2)
    {
        cout << "The maximum number is: " << number3 << endl;
    }
}
```

Write a C++ program to find out the roots of a quadretic equation.

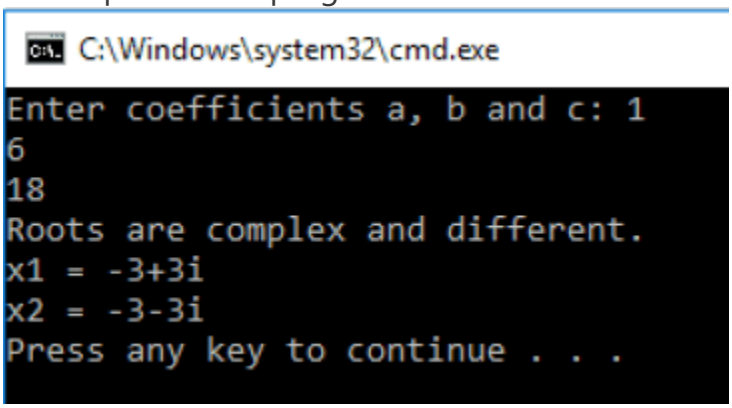
Answer:

```
#include "stdafx.h"
#include <iostream>
#include <iostream>
#include <cmath>
using namespace std;
int main() {
        float a, b, c, x1, x2, discriminant, realPart, imaginaryPart;'
        cout << "Enter coefficients a, b and c: ";'
        cin >> a >> b >> c;
        discriminant = b*b – 4 * a*c;
        if (discriminant > 0) {
            x1 = (-b + sqrt(discriminant)) / (2 * a);
            x2 = (-b – sqrt(discriminant)) / (2 * a);
            cout << "Roots are real and different." << endl;
            cout << "x1 = " << x1 << endl;
            cout << "x2 = " << x2 << endl;
}
else if (discriminant == 0)
{
    cout << "Roots are real and same." << endl;
    x1 = (-b + sqrt(discriminant)) / (2 * a);
    cout << "x1 = x2 =" << x1 << endl;
}
else
{
    realPart = -b / (2 * a);
```

```
    imaginaryPart = sqrt(-discriminant) / (2 * a);
    cout << "Roots are complex and different." << endl;
   cout << "x1 = " << realPart << "+" << imaginaryPart << "i" << endl;
    cout << "x2 = " << realPart << "-" << imaginaryPart << "i" << endl;
}
return 0;
}
```

OUTPUT:

The ouptut of the progarm is as follows



Write a C++ program to find out the area of rectangle and display the result on the screen

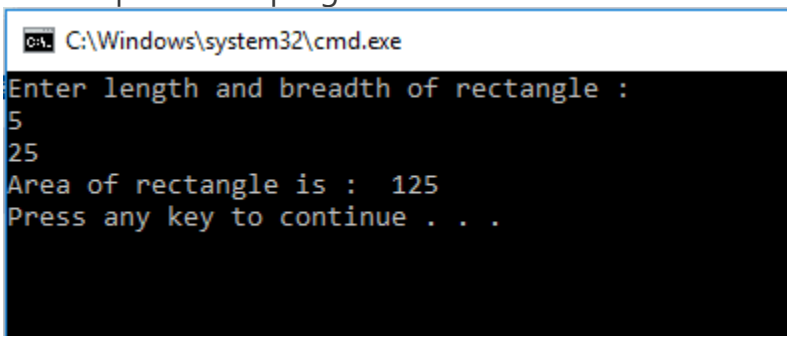
Answer:

```
#include "stdafx.h"
#include <iostream>
#include<conio.h>
using namespace std;
void rectangle(int length, int width)
{
    int area;
    area = length*width;
    cout << "Area of Rectangle is = " << area << "\n\n";
}
int main()//main program
{
    int L, W;
    cout << "Enter LENGTH of rectangle : ";
    cin >> L;
    cout << "Enter WIDTH of rectangle : ";
    cin >> W;
```

```
    rectangle(L, W);
    return 0;
}
```

OUTPUT:

The output of the program is as follows:



Write a C++ program to get the age of a student from the user at run time and display it on the screen.

Answer:

```
#include <stdio.h>
void main() {
    int birthmonth, birthyear;
    int currentmonth, currentyear;
    int agey, agem;
    cout << "\t\t — PROGRAM TO GET AGE –\n\n";
    cout << "Enter Your Birth Year(Eg:1989):";
    cin >> birthyear;
    cout << "\n\nEnter Your Birth Month(Eg:7):";
    cin >> birthmonth;
    cout << "\nEnter The Current Month(Eg:7):";
    cin >> currentmonth;
    cout << "\nEnter The Current Year(Eg:2010):";
    cin >> currentyear;
    agey = currentyear – birthyear;
    agem = 12 – birthmonth;
    cout << "\n\n\t\tYour Age is " << agey << " Years And " << agem << " Months ";
    _getch();
}
```

What is object oriented programming in C++?

OOP stands for Object-Oriented Programming. Procedural programming is about to report procedures or parts that conduct operations on the data, while object-oriented programming is around making objects that have both data and functions.

What is object oriented programming with example?

Each item is said to be an example of a particular class (for example, an object with its name area set to “Mary” might be an instance of class Worker).

Methods in object-oriented programming are understood as methods; variables are also learned as fields, components, attributes, or properties.

Chapter 4: Control Structures

Write a C++ program that takes two integers and characters representing one of the following mathematical operations: +,-,/, or *. Use a switch statement to perform the appropriate mathematical operation on the integers, and display the result. If an invalid operator enters then "Error" message should be displayed and the program should exit (use the exit() function).

Answer:

```
int main()
{
int num1, num2, result;
result = 0;
cout << "Enter First Integer : ";
cin >> num1;
cout << "Enter Second Integer : ";
cin >> num2;
cout << "Choose an from the following \n A-For Addtion \n S-For Subtraction \n D-For Division
\n M-For Multiplication\n";
char opeartion;
cout << "Enter a Character (A, S, D, M): ";
cin >> opeartion;
int i = 0;
switch (opeartion)
{
case 'A':
result = num1 + num2;
cout << "You Selected Addition \nThe Result is = " << result << "\n\n" << endl;
break;
case 'S':
result = num1 – num2;
cout << "You Selected Subtraction \nThe Result is = " << result << "\n\n" << endl;
break;
case 'D':
result = num1 / num2;
cout << "You Selected Division \nThe Result is = " << result << "\n\n" << endl;
break;
case 'M':
result = num1 * num2;
cout << "You Selected Multiplication \nThe Result is = " << result << "\n\n" << endl;
```

```
break;
default:
cout << "Error! operator is not correct";
exit(0);
}
return 0;
}
```

Write a program using for loop that prints even numbers from 0 to 20.

Answer:

PROGRAM USING FOR LOOP THAT PRINTS EVEN NUMBERS FROM 0 T0 20:

```
#include <iostream>
using namespace std;
int main ()
{
int x;
for(x=0 ;x<=20; x=x+1)
if (x%2 ==0)
cout <<"This is an even number"<< x<< ".\n";
}
```

Write a program using while loop that takes an integer for a variable nValue, and returns the sum of all the numbers from 1 to nValue.

Hint: For example, nValue=5 should return 15, which is 1+2+3+4+5

Answer:

```
#include <iostream>
using namespace std;
int main ()
{
int number, i =1, sum =0;
cout<<"Enter a positive integer";
cin>> number;
while (i<= number)
{
```

```
sum +=i;
++i;
}
cout<<"Sum of " << number <<"="<< sum;

return 0;
}
```

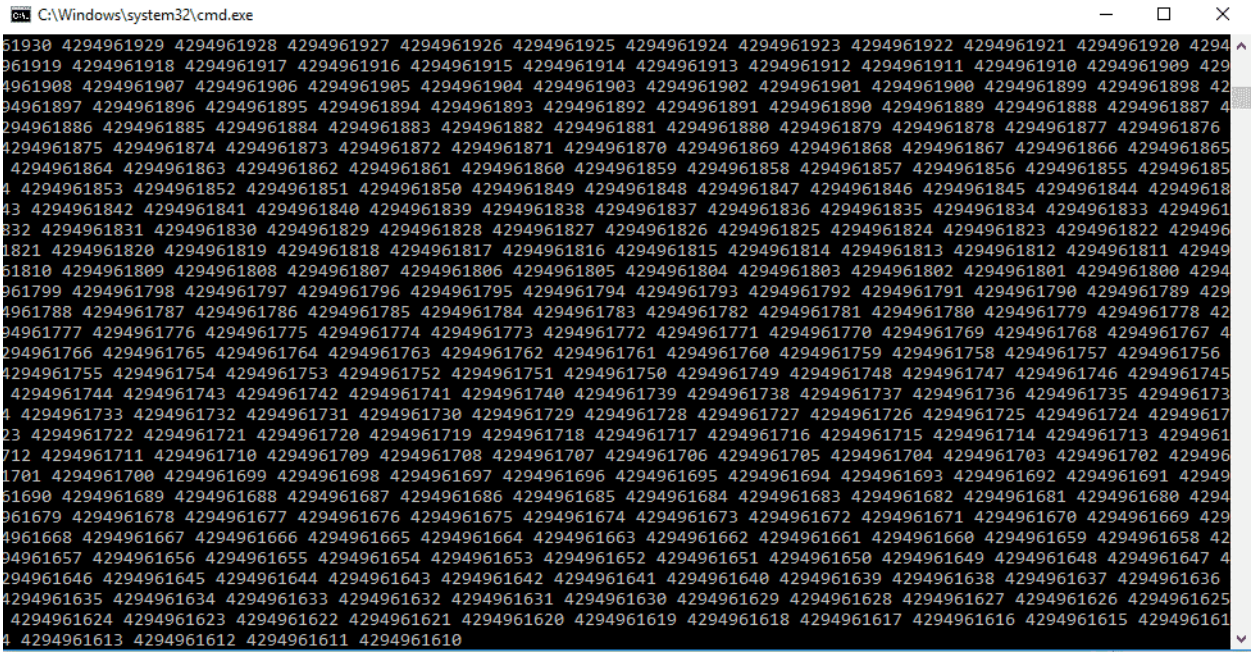
What is wrong with the following for loop?

```
// Print all numbers from 9 to 0
for (unsigned int nCount = 9; nCount >=0; nCount –)
cout <<nCount <<" ";
```

Answer:

PROBLEM:

The problem in the loop is that an **unsigned number** is never negative.
Therefore, the loop-test **nCount >= 0** will always be true. Thus you get an infinite loop.
The output of the infinite loop with the above mentioned code is as follows:



Write a C++ program to read the address of a person and exit when the user enters dot (.) from the keyboard.

Answer:

```
#include <iostream>
using namespace std;
int main()
{
char ch;
cout << "Enter address : " << endl;
ch = _getche();
while (ch != '.')
{
ch = _getche();
}
cout << "\n\n\n";
cout << "You pressed dot(.)" <<"\n\n\n";
return 0;
}
```

Write a C++ program to find out the area of a triangle and if any side is zero then display the message "There is no triangle".

Answer:

```
#include<iostream>
using namespace std;
int main ()
{
int height, base;
float ans;
cout <<"Enter height and base:";
cin>>height>> base;
ans=(0.5)*height*base;
if (height == 0 ||base==0)
cout<< "there is no triangle";
else
cout<<"Area of triangle is :<<ans;
return 0;
}
```

Write a C++ program to input a character from the keyboard and display the message after testing whether it is Vowel or Consonant.

Answer:

PROGRAM TO CHECK WHETHER ENTERED CHARACTER IS VOWEL OR CONSONANT:

```
#include<iostream>
using namespace std;
int main ()
{
char c;
int isLowercaseVowel , isUppercaseVowel;
cout<<"Enter an alphabet:";
cin>> c;
isLowercaseVowel =( c== 'a' || c=='e' || c=='i' || c=='o' ||c=='u');
isUppercaseVowel =( c=='A' || c=='E' || c=='I' || c== 'O' || c=='U');
if (isLowercaseVowel || isUppercaseVowel)
cout<<c<< "     is a Vowel";
else
cout<< c<< "    is a consonant";
return 0;
}
```

Chapter 5: Arrays and Strings

Write down a C++ program to find the multiplication of two matrices A[3][2] and B[2][3] .

Answer:

```
int main()
{
   int r1, c1, r2, c2, i, j, k;
   int A[3][2], B[2][3], C[3][3];
   cout << "Enter elements of matrix A : ";
   for (i = 0; i < 3; i++)
   for (j = 0; j < 2; j++)
      cin >> A[i][j];
   cout << "Enter elements of matrix B : ";
   for (i = 0; i < 2; i++)
   for (j = 0; j < 3; j++)
      cin >> B[i][j];
   for (i = 0; i < 3; i++)
   {
      for (j = 0; j < 3; j++)
      {
         C[i][j] = 0;
         for (k = 0; k < 2; k++)
         {
            C[i][j] += A[i][k] * B[k][j];
         }
      }
   }
   cout << "Product of matrices\n";
   for (i = 0; i < 3; i++)
   {
      for (j = 0; j < 3; j++)
         cout << C[i][j] << "  ";
      cout << "\n";
   }
   return 0;
}
```

Write down a C++ program to find the subtraction and addition of two matrices A[3][3] and B[3][3].

Answer:

```
int main()
{
   int i, j;
   int A[3][3], B[3][3], Add[3][3], Sub[3][3];
   cout << "Enter elements of matrix A : ";
   for (i = 0; i < 3; i++)
   for (j = 0; j < 3; j++)
      cin >> A[i][j];
   cout << "Enter elements of matrix B : ";
   for (i = 0; i < 3; i++)
   for (j = 0; j < 3; j++)
      cin >> B[i][j];
   for (i = 0; i < 3; i++)
   {
      for (j = 0; j < 3; j++)
      {
         Add[i][j] = 0;
         Add[i][j] += A[i][j] + B[i][j];
      }
   }
   for (i = 0; i < 3; i++)
   {
      for (j = 0; j < 3; j++)
      {
         Sub[i][j] = 0;
         Sub[i][j] += A[i][j] – B[i][j];
      }
   }
   cout << "Addition of matrices\n";
   for (i = 0; i < 3; i++)
   {
      for (j = 0; j < 3; j++)
         cout << Add[i][j] << "  ";
      cout << "\n";
   }
   cout << "Subtraction of matrices\n";
```

```
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 3; j++)
            cout << Sub[i][j] << "  ";
        cout << "\n";
    }
    return 0;
}
```

Write a C++ program to find the transpose of a matrix A[3][3].

Answer:

```
int main()
{
    int A[3][3], m, n, i, j;
    cout << "Enter elements of matrix : ";
    for (i = 0; i < 3; i++)
    for (j = 0; j < 3; j++)
        cin >> A[i][j];
    cout << "Entered Matrix : \n ";
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 3; j++)
            cout << A[i][j] << " ";
        cout << "\n ";
    }
    cout << "Transpose of Matrix : \n ";
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < m; j++)
            cout << A[j][i] << " ";
        cout << "\n ";
    }
    return 0;
}
```

Write a C++ program to read the temperature of the

whole week in an array and then find the hottest day of the week.

Answer:

```
int main()
{
    float days = 0;
    float temperatures[50];
    float temptotal = 0;
    float average = 0;
    float min = 9999999999999;
    float max = -9999999999999;
    string dayName;
    for (int i = 0; i < 7; i++)
    {
        cout << "Enter the temperature for day number " << i+1 << " : ";
        cin >> temperatures[i];
        temptotal += temperatures[i];
       if (temperatures[i] > max)
        {
            max = temperatures[i];
        }
        if (temperatures[i] < min)
            min = temperatures[i];
    }
    average = (temptotal / days);
    cout << "The Hottest Day is having the Temperature : "<< max << endl;
    return 0;
}
```

Write a C++ program to find sum of the values of a two dimensional array int Test [2][3] and display the result on the screen.

Answer:

```
int main()
{
   int i, j, k;
   int A[2][3], Add[2][3], C[3][3];
   cout << "Enter elements of matrix A : ";
   for (i  i < 2; i++)
   for (j = 0; j < 3; j++)
      cin >> A[i][j];
   for (i = 0; i < 2; i++)
   {
      for (j = 0; j < 3; j++)
      {
         Add[i][j] = 0;
         Add[i][j] += A[i][j] + A[i][j];
      }
   }
   cout << "Addition of matrices\n";
   for (i = 0; i < 2; i++)
   {
      for (j = 0; j < 3; j++)
         cout << Add[i][j] << "        ";
      cout << "\n";
   }
   return 0;
}
```

Chapter 6:Functions

What is a function?

Ans. In structured programming the program consists of more than more one part. Each part of program is called a module or function. Every function is given a unique name and it is developed to perform a specific task. So function can be defined as " A named piece of code developed to perform a specific task is called function".

Why functions are used?

Ans. Function is a piece of code designed to perform a specific task. There are many advantages of using functions. These advantages are described below:

- Easy programming
- Easy modification
- Easy debugging
- Reuse-ability
- Eliminates duplicate code
- Less programming time

What are built-in functions?

Ans. The function that are provided as a part of C language are called built-in functions. These functions are also called library function. A large number of built-in functions are provided by C language. These functions are stored in different header files. If we want to use a built-in function in a program the relevant header files is included at the start of the program in Preprocessor directive.

What are user defined functions?

Ans. The functions that are written by the programmer to perform specific task are

called user defined functions. These functions are written according to the requirement of the program.

What is function prototypes?

Ans. Function declaration is also called function prototype. It is a statement that provides basic information to compiler about the structure of the function like other C language statement, function declaration statement also ends with semicolon. Function declaration is necessary like variable declaration. A function must be declared in a C language program. Function can be declared before the main() function or inside the main() function.

What is function definition?

Ans. Every function perform some specific task. The task is performed when the set of instructions execute. Writing set of statements of a function is called function definition. Function definition is always done outside main() function.

What is function header?

Ans. The first line of the function definition is called function header. Its general syntax is as follow:

• Return-Type Name(parameters)

What is function calling?

Ans. The statement that is written to use a function is called function call. A function can be called at any point in the program. A function is called by using its name. The required parameters are maintained after the name in braces at the end of the function call statement.

What is return statement?

Ans. Keyword "return" is used to return a value from the body of called function to calling function. The statement in which "return" keyword is used is called return statement. The general syntax for return statement is as follow:

return expression;

What are parameters?

Ans. Parameters are also called arguments. These are the values that are provided toa function when it is called. when function is called parameters are written after function name in parenthesis. These parameters can be variables or constants. More than one parameter is separated by comma.

What is a local variable?

Ans. The variables declared inside main() function, inside any user defined function or header of function definition are called local variables. Local variable also called automatic variables. The general syntax to declare a local variable is as follows:

auto data-type variables-name;

What is global variable?

Ans. The variables that are declared outside the main() function or any other function are called global variable. Global variables are also called external variables. Global variables can be used by all functions in the program. All functions can share their value.

If value of a global variable is changes in a function, that changes value is also available in other functions.

What is meant by life time of a variable?

Ans. Lifetime of a local variable is limited, when control enters into the function and variable declaration statement is executed, they are created in memory. When the control exits from the function these variables are destroyed and their life ends, when variables are destroyed the data stored in them also becomes inaccessible.

What is meant by scope of a variable?

Ans. Local variables have a limited scope they can only be used in the function in

which they are declared. Compiler generates an error if we want to access a local variable, outside its scope.

What is scope of global variable?

Ans. Global variables can be accessed in all modules of program. They are accessible in main() function as well as all other user defined functions.

What is life time of global variable?

Ans. When program starts execution, global variables are created in memory. They remain in memory till the termination of the program. When the program is terminated global variables are destroyed from the memory. Therefore life time of a global variable is between starting and termination of program.

(Learn the remaining topics from text book)

Chapter 8: Objects and Classes

Write a C++ program implementing a class with the name Circle having two functions with the names: GatRadius and CalArea. The functions should get the value for the radius from the user and then calculate the area of the circle and display the result.

Answer:

```
class Circle
{
public:
	int GetRadius()
	{
		int radius;
		cout << "Enter Radius of the circle :";
		cin >> radius;
		return radius;
	}
	double CalArea(double p, double radius)
	{
		double area = p * radius * radius;
		return area;
	}
};

int main()
{
	Circle c;
	int r = c.GetRadius();
	double p = 3.14;
	cout << "Area of Circle is : " << c.CalArea(p, r) << endl;
	return 0;
}
```

Write a C++ program implementing inheritance between Employee (base class) and Manager (derived class).

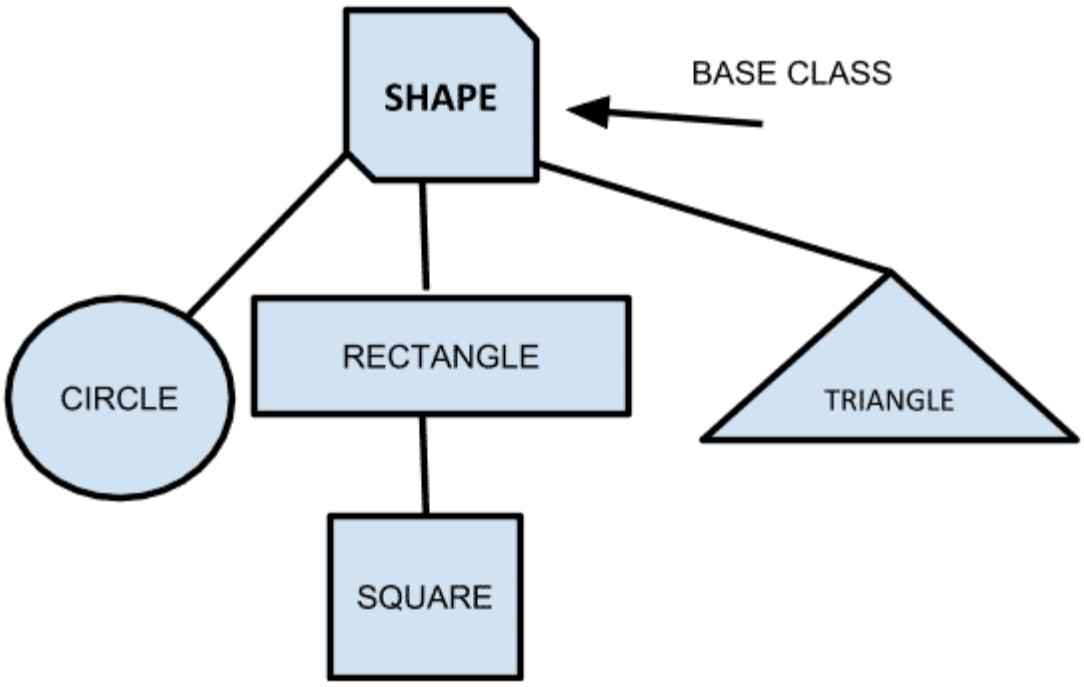
Answer:

```
class Employee
{
public:
	void getBasicInfo()
	{
		cout << "– BASIC INFORMATION –\n";
		cout << "Name    : Adeel Raheem.";
		cout << endl;
		cout << "Emp. Id : 2206";
		cout << endl;
		cout << "Gender  : M";
		cout << endl;
	}

	void getDepartment()
	{
		string department;
		cout << "– DEPT. INFORMATION –\n";
		cout << "Department Name : HR\n";
		cout << "Designation     : Manager\n";
		cout << "Working Hours   : 16 hrs/day\n\n";
	}
};

class Manager : public Employee
{
public:
	void display()
	{
		getBasicInfo();
		cout << "\n\n\n\n";
		getDepartment();
	}
};
	int main()
	{
		char c;
		Manager m;
```

```
        cout << "Do you want to get Employee Information? Select Y/N :";
        cin >> c;
        cout << "\n\n";
        if (c == 'Y')
        {
            m.display();
        }
        else if (c == 'N')
        {
            cout << "\nNothing To Display";
        }
        else
        {
            cout << "\nInvalid Selection";
        }
}
```

Diagrammatically represent the following scenario of inheritance:

- We have **Shape** as a base class and **Circle**, **Rectangle**, **Triangle** and **square** as its derived classes.



Write a C++ program implementing a class with the name ConstDest having constructor and destructor functions in its body.

Answer:

```
class ConstDest
{
public:
ConstDest()
{
cout << "I am a Constructor" << endl;
}
~ConstDest()
{
cout << "I am a Destructor" << endl;
}
};
int main()
{
ConstDest c1;
_getch();
return 0;
```

```
}
```

Write a C++ program implementing a class with the name Time. This class should have a constructor to initialize the time, hours, minutes and seconds. The class should have another function named ToSecond() to convert and display the time into seconds.

Answer:

```
class Time
{
   private:
      int seconds;
      int hh, mm, ss;
   public:
         void getTime(void);
         void convertIntoSeconds(void);
         void displayTime(void);
};

void Time::getTime(void)
{
         cout << "Enter time:" << endl;
         cout << "Hours?   ";         cin >> hh;
         cout << "Minutes? ";         cin >> mm;
         cout << "Seconds? ";         cin >> ss;
}

void Time::convertIntoSeconds(void)
{
         seconds = hh * 3600 + mm * 60 + ss;
}

void Time::displayTime(void)
{
         cout << "Time in total seconds: " << seconds<<"\n\n";
}

int main()
{
         Time T;
```

```
        T.getTime();
        T.convertIntoSeconds();
        T.displayTime();

        return 0;
}
```

What is object? How objects are created to access members of a class?

Answer:
OBJECT
DEFINITION:
"Objects are instances of the class, which holds the data variables declared in the class and the member functions work on these class objects. In other words, a variable of type class is called an object."
EXPLANATION:
In C++, when we declare a variable of type class, we call it instantiating (from the instance,) the class and the variable itself is called an instance or object of that class. The object is of great importance in OOP, because, a class cannot be used without creating its object.
GENERAL FORM:
The general syntax for creating an object of a class is given below.
Class_name Object_name;
Thus, for the above class CRectangle, the object can be created as follows:
CRectangle R1; II R1 is the object of class CRectangle
We can also create more than one object in a single statement like:
CRectangle R1, R2, R3;
EXPLANATION OF GENERAL FORM:
Here, R1, R2, R3 are the objects of the same class CRectangle that share the same data member and member functions. The definition of a class does not occupy any memory. It only defines what the class looks like. In order to use a class, a variable of that class type must be declared. When an object is created then memory is set aside for all the data members and member functions of that class.
EXAMPLE:
Consider the following program to implement the class CRectangle discussed above.

```
#include "stdafx.h"
#include <iostream>
#include<conio.h>
using namespace std;
```

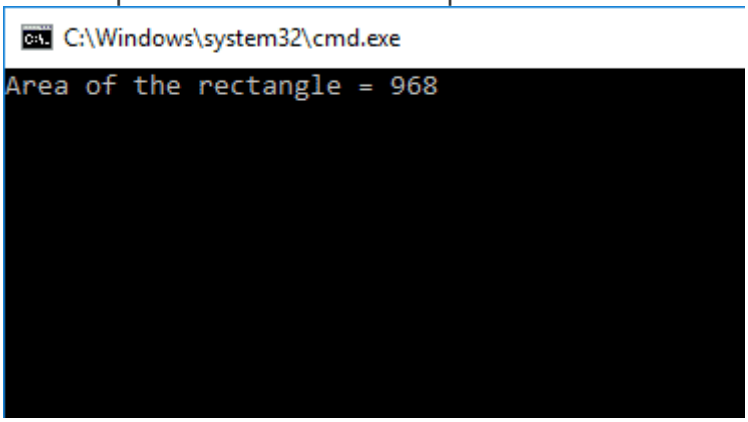
```
class CRectangle
{
private:
   int X, Y;
public:
   void set_values(int a, int b)
   {
      X = a;
      Y = b;
   }
   int area()
   {
      cout << "Area of the rectangle = " << (X*Y);
      return 0;
   }
}; //End of class
int main()
{
   CRectangle R1; // Object creation
   R1.set_values(44, 22); //call to set_values function
   R1.area(); // call to area function
   _getch();
   return 0;
}
```

EXPLANATION OF PROGRAM:

In this program the member function set-values and area are accessed by R1 .set-values(44,22) and R.area().

OUTPUT:

The output of the above example is as follows:



Inheritance and polymorphism with examples.

Answer:
Inheritance
Definition:
"Acquiring some of the common qualities from parents (for instance eyes like mother, hair-like father, etc) is called inheritance."
A new model car having some inherited features from the old model, like the brake system and navigation system, etc.
Example:

```
class Employee
{
public:
	void getBasicInfo()
	{
		cout << "– BASIC INFORMATION –\n";
		cout << "Name    : Adeel Raheem.";
		cout << endl;
		cout << "Emp. Id : 2206";
		cout << endl;
		cout << "Gender  : M";
		cout << endl;
	}

	void getDepartment()
	{
		string department;
		cout << "– DEPT. INFORMATION –\n";
		cout << "Department Name : HR\n";
		cout << "Designation     : Manager\n";
		cout << "Working Hours   : 16 hrs/day\n\n";
	}
};

class Manager : public Employee
{
public:
	void display()
	{
		getBasicInfo();
		cout << "\n\n\n\n";
		getDepartment();
	}
};
```

```
int main()
{
        char c;
        Manager m;
        cout << "Do you want to get Employee Information? Select Y/N :";
        cin >> c;
        cout << "\n\n";
        if (c == 'Y')
        {
                m.display();
        }
        else if (c == 'N')
        {
                cout << "\nNothing To Display";
        }
        else
        {
                cout << "\nInvalid Selection";
        }
}
```

Polymorphism

Definition:

"It is the ability to use an operator or function in multiple ways."

It means multiple possible states for a single property. For example, a person can be a student as well as a friend to another person. Also, a person can be an engineer as well as a teacher.

In C++ polymorphism can be achieved by one of the following concepts:

- Function overloading
- Operator overloading
- Virtual functions

Example:

Consider the following example:

6 + 10;

The above statement refers to integer addition with the help of '+' operator. The same operator can be used for adding the floating point numbers. Or it can be used for concatenation of strings. They are as follows:

7.15 + 3.78

"Computer" + "Training"

The above concept is known as operating overloading.