OBJECT ORIENTED PROGRAMMING WITH C++ -18BIT23C

TEXT BOOK:

“Object oriented programming with ANSI and Turbo C++” Pearson Education publications, 2006 .Ashok N Kamthane.

PREPARED BY DR.M.SORANAMAGESWARI

Core: OBJECT ORIENTED PROGRAMMINGWITH C++
II Semester -Subject Code : 18BIT23C

UNIT I: Introduction to C++: Key concepts of OOPs – Advantages – object oriented languages – Inputand output in C++: Streams in C++ - Pre-Defined Streams – Unformatted console I/O operation –Formatted console I/O operations – C++Declarations – Control structures: Decision Making statements –If….Else – Jump – GOTO – Break – Continue – Switch case statements – Loops in C++ : For – While – Do… While Loops – Functions in C++ - In Line Functions – Function Overloading.

UNIT II: Class and Object: Declaring objects – Defining Member Functions – Static Member Variablesand Functions – Array of Object – Friend Functions – Overloading Member Functions – Bit Fields andClass. Constructor and Destructors: Characteristics – Calling Constructor and Destructors – Constructorsand Destructors with Static Member.

UNIT III: Operator Overloading: Overloading Unary – Binary Operators – Overloading FriendFunctions – Type Conversion – Inheritance: Types of Inheritance – Single – Multilevel – Multiple –Hierarchical – Hybrid and Multi Path Inheritance – Virtual Base Classes – Abstract Classes.

UNIT IV: Pointers: Declaration – Pointer to Class – Object – THIS Pointer – Pointer to Derived Classes and Base Classes – Arrays: Characteristics – Arrays of Classes – Memory Models – New and Delete Operators – Dynamic Object – Binding – Polymorphisms and Virtual Functions.

UNIT V: Files: File Stream Classes – File Modes – Sequential Read/ Write Operations – Binary andASCII Files – Random Access Operation – Command Line Arguments - Exception Handlings :Principles of Exception Handling – The Keywords try, Throws and Catch – Exception Handling Mechanism – Multiple Catch Statements – Catching Multiple Exceptions – Re-throwing Exception –Strings: Declaring and Initializing String Objects – Strings Attributes – Miscellaneous Functions.

TEXT BOOK

1. Ashok N Kamthane, "Object Oriented Programming with ANSI and Turbo C++", Pearson Education Publications, 2006.

REFERENCE BOOKS

1. Balagurusamy.E, Object Oriented Programming with C++ - TMH Pub 1998.
2. John R Hubbard, Programming with C++ - TMH Publ. II Edition 2002.

Object oriented programming with C++

UNIT-I:

Introduction to C++: Key concepts of OOPs – Advantages – object oriented languages – Input and output in C++ -Streams in C++ - Pre-Defined Streams – Unformatted console I/O operation –Formatted console I/O operations – C++ Declarations – Control structures: Decision Making statements–If....Else – Jump – GOTO – Break – Continue – Switch case statements – Loops in C++ : For – While – Do... While Loops – Functions in C++ - In Line Functions – Function Overloading.

Introduction of C++

Key concepts of OOPS:

- Objects
- Classes
- Methods
- Data Abstraction
- Encapsulation
- Inheritance
- Polymorphism
- Dynamic Binding
- Message Passing
- Reusability
- Delagation
- Genericity

- **Objects:**

Objects are the basic run time entities in an object-oriented system. They may represent a person, a place, a bank account, a table of data or any item that the program has to handle. They may also represent user-defined data such as vectors, time and lists.

- **Classes :**

The entire set of data and code of an object can be made a user-defined data type with the help of class. In fact, objects are variables of the type class. Once a class has been defined, we can create any number of objects belonging to that class. Each object is associated with the data of type class with which they are created. A class is thus a collection of objects similar types.

For examples, Mango, Apple and orange members of class fruit. Classes are user-defined that types and behave like the built-in types of a programming language.

- **Methods:**

- ❑ Operation required for an object entity coded in a class.
- ❑ operation defined in a class.
- ❑ Action are defined in a method.
- ❑ Class contains private & public methods or number function.
- ❑ Member function contains data members.

(e.g)

```
class student
{
Private:
Data members;
Public:
Method()
{….}
};
void  main()
```

```
{
student s
s.method name();
}
```

- **Data Abstraction:**

 Abstraction refers to the act of representing essential features without including the background details or explanation.

(e.g) size,cost,height,weight.

- **Encapsulation :**

 The wrapping up of data and function into a single unit (called class) is known as encapsulation. Data and encapsulation is the most striking feature of a class. The data is not accessible to the outside world, and only those functions which are wrapped in the class can access it.

- **Inheritance :**

 Inheritance is the process by which objects of one class acquired the properties of objects of another classes. It supports the concept of hierarchical classification.

- **Polymorphism:**

Polymorphism is another important OOP concept. Polymorphism, a Greek term, means the ability to take more than on form. An operation may exhibit different behavior is different instances. The behavior depends upon the types of data used in the operation.

- **Dynamic Binding**:

Binding refers to the linking of a procedure call to the code to be executed in response to the call. Dynamic binding means that the code associated with a given procedure call is not known until the time of the call at run time. It is associated with polymorphism and inheritance. A function call associated with a polymorphic reference depends on the dynamic type of that reference.

- **Message Passing:**

Objects communicate with one another by sending and receiving information much the same way as people pass messages to one another.

- **Reusabality:**

  classes can be used by other classes.

- **Delegation:**

  Provided reusability based on the relationship**.**

- **Genericity:**

  More than one version.

Advantages of OOP:

- OOP offers several benefits to both the program designer and the user. Object Orientation contributes to the solution of many problems associated with the development and quality of software products. The new technology promises greater programmer productivity, better quality of software and lesser maintenance cost. The principal advantages are:
- Through inheritance, we can eliminate redundant code extend the use of existing Classes.

- The principle of data hiding helps the programmer to build secure program that can not be invaded by code in other parts of a programs.
- All objects oriented programming language can create returned and reusable parts of programs.

Input and output in C++

-Stream is a flow of data in bytes in sequence.

-If input data is received from input device in sequence is called as source stream.

-When the data is passed to output devices then its called Destination stream.

Predefined streams:

cin - standard I/P

cout - standard O/P

clog - control error messages

- **I/P Functions**:

cin	**cout**
✓get()	put()
✓getline()	write()
✓read()	flush()
✓peck()	
✓ignore()	
✓g cout	

Sample programs:

1).

```
#include<iostream.h>
#include<conio.h>
Void main ()
{
Clrscr();
cout<<“Hello<<“\n”;
cout<<“c++”;
getch();
}
```

2).

```
#include<iostream.h>
#include<conio.h>
void main ()
{
clrscr():
int a=10;
int b=10;
int c;
c = a+b;
cout<<c;
getch();
}
```

3).Division of 2 numbers:

```
#include<iostream.h>
#include<conio.h>
Void main ()
{
clrscr();
int a,b,c;
cout<<“Enter the value of a=“ ;
cin>>a;
cout<<“Enter the value of b=“ ;
cin>>b;
c=a/b;
cout<<“Division  of  two no.   =” ;
cout<<c;
getch()
}
```

Output:

```
Enter the value of a=20
Enter the value of b=10
Division of two no. =2
```

4).Average of 2 numbers:

```
#include<iostream.h>
#include<conio.h>
Void main ()
{
clrscr();
int a,b;
Float c;
cout<<“Enter the value of a=“ ;
cin>>a;
cout<<“Enter the value of b=“ ;
cin>>b;
c=(a+b)/2 ;
cout<<“Average  of a and b =”;
cout<<c;
getch();
}
```

```
Enter the value of a=100
Enter the value of b=50
Average  of a and b =75.0
```

5).Display the Name

```
#include<iostream.h>
#include<conio.h>
void main ()
{
char name[10];
clrscr();
cout<<“Enter the name”;
cin>>name;
cout<<“Your name is”<<name”;
getch();
}
```

Output:

```
Enter the name kalam
Your name is kalam
```

Formatted console I/O Operations:

c++ supports a number of features that could be used for formatting the output. The features include:

- width()
- precision()
- fill()
- setf()
- unsetf()

width(): To specify the required size for displaying an output value

precision(): To specify the number of digits to be displayed after the decimal point of a float value.

fill(): To specify a character that is used to fill the unused portion of filled.

setf(): The setf() method is used to set various flags for formatting output.

unsetf() : The unsetf() method is used to remove the flag setting.

- **Setting Precision: precision() :**

We can specify the number of digits to be displayed after the decimal point while printing the floating point numbers.

Syntax :

```
cout.precision(d);
```

Where ‘ d’ is the number of digits to the right of the decimal point.

- **Filling and Padding: fill()**

The unused positions of the field are filled with white spaces .However, the fill() function can be used to fill the unused positions.Where character represents the character which is used for filling the unused positions.

setf() :

The setf() member function of the ios class is used for various types of formatting.

Syntax:

```
cout<<setf(arg1, arg2);
```

- The argument is one of the formatting flags, specifying the action required for the output.
- The argument known as bit field which specifies the group to which the formatting flag belongs.
- There are three bit fields and each has a group of format flags.

Formatting Flags, Bit-fields and setf():

Consider the following segment of code:

```
cout.fill('*');
cout.setf(ios: :left, ios::adjustfield);
cout. width(15);
cout<<“TABLE 1” << “/n”;
```

Will produce the following output:

```
TABLE 1 * * * * * * * * * * *
```

flag	meaning
ios :: showbase	Use base indicator on output
ios :: showpas	Print + before positive number
ios :: showpoint	Show trailing decimal point and zeros
ios :: uppercase	use uppercase letters for hex output
ios :: skipus	skip white space on input
ios :: unitbuf	flush all streams after insertion
ios :: studio	flush stdout and stderr after insertion

- **Unformatted 1/O Operations :**

***** cin and cout are the two predefined streams of input and output of data.The operator << (insertion operator)is overloaded in the output stream and >> (Extraction operator) operator is overloaded in input stream.unformatted functions are as follows

* put() *get() * getline() *read() * write() functions

put() and get () Functions :

- The classes istream and ostream define two member functions get() and put() to handle the single character input and output operations**.**
- There are two types of get() functions:

  get (char)

  get (void).
- get(char) version assigns the input character to its argument.
- get(void) version returns the input character.

getline() and write () Functions :

- The getline() function reads a whole line of text that ends with a newline character.
- This function can be invoked by using the object cin.

```
cin.getline (line, size);
cin.getline(name,10);
```

- The function getline() which reads character input name with the size10.
- The reading is determinated as soon as either the new line character is read or size-1 characters are read.

Parts of the c++ program is as follows

- Include fields
- Class definition or declaration
- Class function definition
- Main() function

TOKENS:

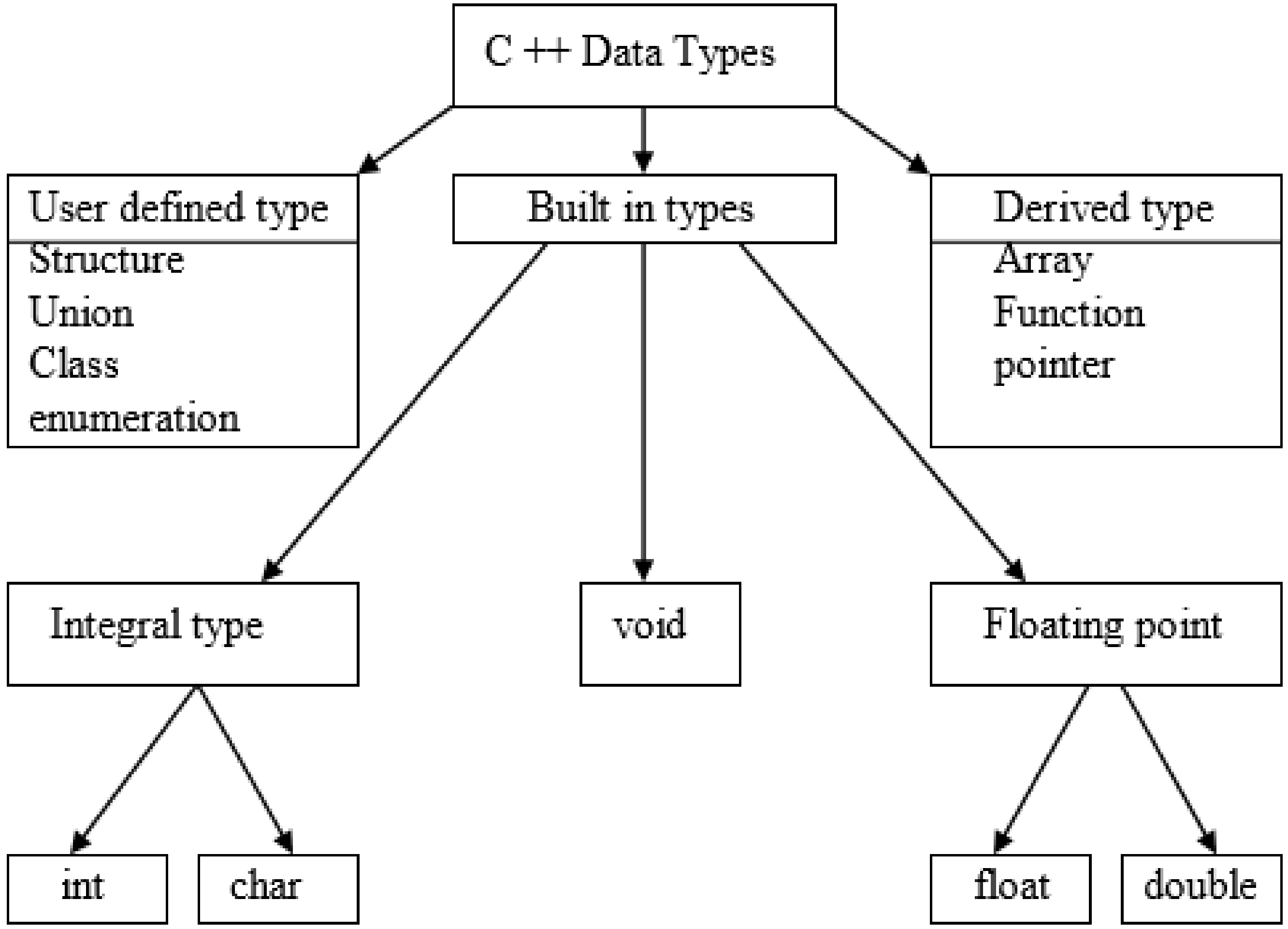
The smallest individual units in program are known as tokens. C++ has the following tokens.

i. Keywords ii. Identifiers iii. Constants iv. Strings v. Operators

C ++ Data Types :

Both C and C++ compilers support all the built in types. With the exception of void the basic datat ypes may have several modifiers preceding them to serve the needs of various situations. The modifiers signed, unsigned, long and short may applied to character and integer basic data types. However the modifier long may also be applied to double.

BASIC DATA TYPES IN C++



Data types in C++ can be classified under various categories.

TYPE	**BYTES**	**RANGE**
char	1	-128 to – 127
usigned	1	0 to 265
sgned char	1	-128 to 127
int	2	-32768 to 32768
unsigned int	2	0 to 65535
singed int	2	-32768 to 32768
short int	2	-32768 to 32768

long int	4	-2147483648 to 2147483648
signed long int	4	-2147483648 to 2147483648
unsigned long int	4	0 to 4294967295
float	4	3.4E-38 to 3.4E+38
double	8	1.7E -308 to 1.7E +308
long double	10	3.4E-4932 to 1.1E+ 4932

- **USER DEFINED DATA TYPES:**

- **STRUCTURE AND CLASSES**

- We have used user defined data types such as structure and union in C. While these more features have been added to make them suitable for object oriented programming. C++ also permits us to define another user defined data type known as class which can be used just like any other basic data type to declare a variable. The class variables are known as objects, which are the central focus of OOPs.

ENUMERATED DATA TYPE:

- An enumerated data type is another user defined type which provides a way for attaching names to number, these by increasing comprehensibility of the code. The enum keyword automatically enumerates a list of words by assigning them values 0,1,2 and soon. This facility provides an alternative means for creating symbolic.

- **CONTROL STRUCTURES**:

-Like c, c++ supports all the basic control structures and implements them various control statements.

- **Decision making statements**:

The if statement:

The if statement is implemented in two forms:

1. simple if statement
2. if… else statement

Simple if statement:

```
if (condition)
{
action;
}
```

If.. else statement

```
if (condition)
  Statment1;
else
  Statement2;
```

The switch statement

This is a multiple-branching statement where, based on a condition, the control is transferred to one of the many possible points;

```
Switch(expr)
{
case 1:
action1;
Break;
case 2:
action2;
break;
..
..
default:
message
}
```

Jumping Statements:

The jump statement unconditionally transfer program control within a function. C language provides us multiple statements through which we can transfer the control anywhere in the program.

There are basically 3 Jumping statements:

- Jumping statements
- continue statement
- goto statement

Break satement:

-Sometimes, it s necessary to exit immediately from a loop as soon as the conduct satisfied

-The break statement terminates the execution of the enclosing loop or statement. In a loop statement, the break statement can stop the counting when given condition becomes true.

-The break statement is a jump instruction and can be used inside switch construct, for loop. while loop and do-while loop.

-The execution of break statement causes immediate exit from the concern construct and the control is transferred to the statement following the loop.

-In the loop construct the execution statement further execution of the program is reserved with the terminates loop and further execution of the program reserved with the statement showing the day .

Syntax:

The syntax for a break statement in C is as follows:

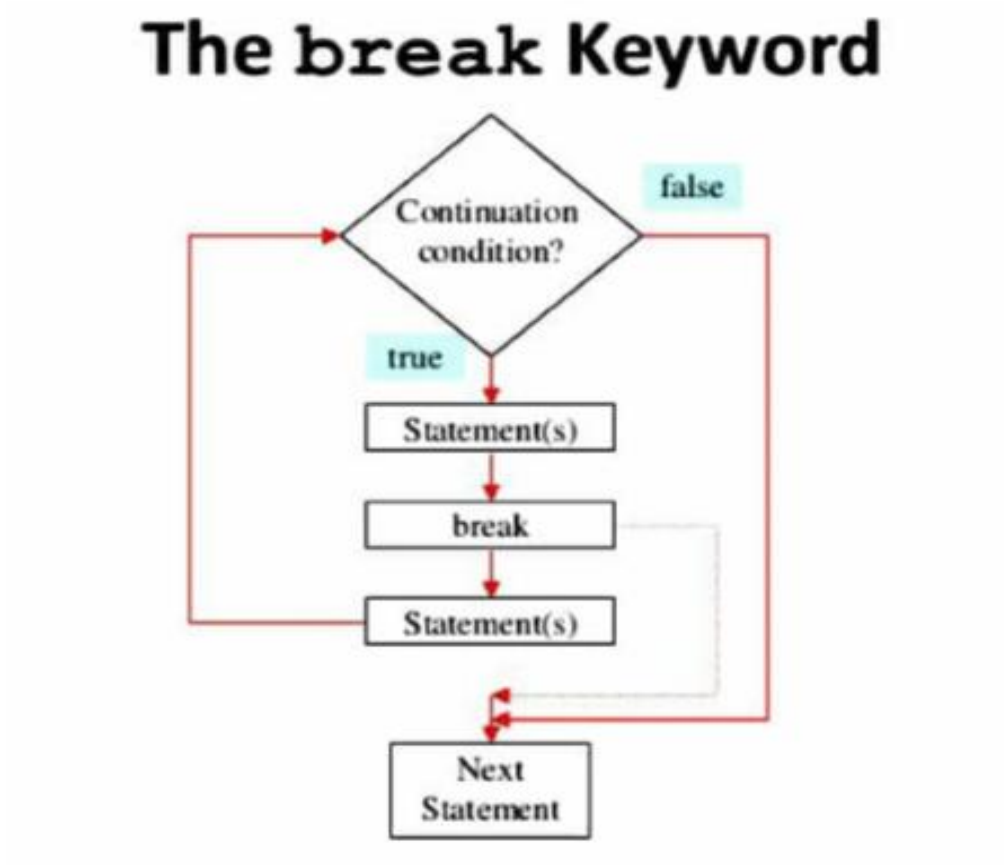
Break;

The break statement in the programming language has the following two usages:

1. When the break statement is encountered inside a loop. the loop is immediately terminated and program control resumes at the next statement following the loop
2. can be used to terminate a case in the switch statement.

```
#include <stdio.h>
#include <conio.h>
void main()
{
```

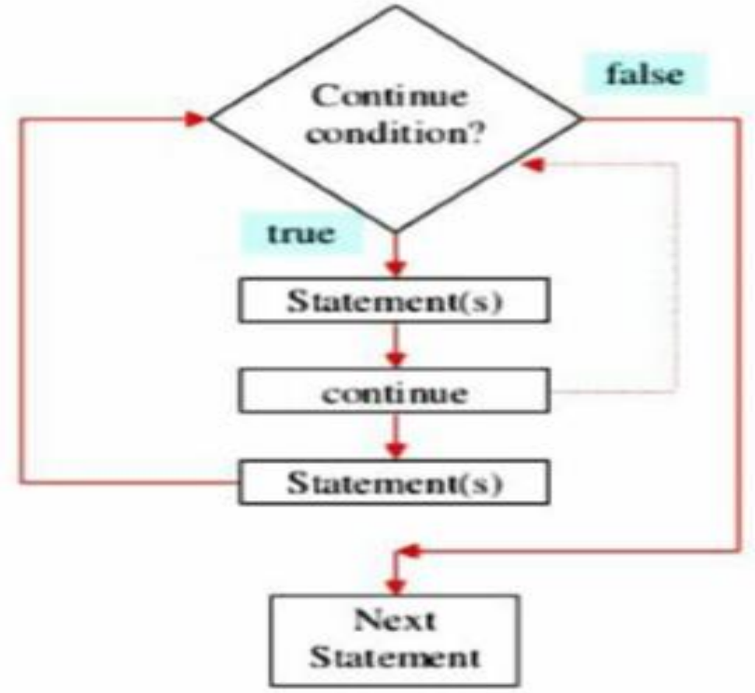
```
int ;
for(i-0; i<100; i++)
{
If i==10) break; // terminate loop if i is 10
printf("i: %d \n", i);
{
printf "Loop complete.");
}
```



Continue :

- The continue statement provides a convenient way to force an immediate jump to the loop control statement. The break statement terminates the execution of the loop.
- As you can see in the given example, we have used both the statements within the do while loop. The program prompts the user to entire any number. after number is less than 0, the break statement terminates the execution of the loop. If the number is greater than 10, the continue statement
- skips the value and jumps to the do while loop without terminating the loop. Otherwise, it will print the entered number.

The `continue` Keyword



- **continue :**

1.Skips the remaining statements in the body of a while, for or **do/while** structure .Proceeds with the next iteration of the loop

2**.while and do/while** Loop-continuation test is evaluated immediately after the continue statement is executed .

3. **for structure** Increment expression is executed, then the loop-continuation test is evaluated .

goto Statement:

It is a welknown as jumping statement.It is primarily used to transfer the control of execution to any place in a program. It is useful to provide branching within a loop.

- When the loops are deeply nested at that if an error occurs then it is difficult to get exited from such loops. Simple break statement cannot work well properly. ln this situations. goto statements used. A goto statement in C programming language provides an unconditional jump from the goto lo a labeled statement in the same function.

Syntax:

```
goto Label;
…
…
Label: statement;
```

```
#include <studio.h>
#include <conio.h>
void main()
{
int a;
start :
printf (“enter any no.”);
scanf( “%d”,  &a);
if(a<0);
goto start; //Goto statement A
a=a*a;
printf(“\n %d", a);
getch();
}
```

- **Loops in C++**
- For
- While
- Do while

For loop:

Syntax:

```
for (initialization; condition; update)
 {
//body of-loop
 }
```

- initialization - initializes variables and is executed only once
- condition - if true, the body of for loop is executed if false, the for loop is terminated
- update - updates the value of initialized variables and again checks the condition

- **Example:**

```
#include <iostream>
using namespace std;
int main()
 {
 for (int i = 1; i <= 5; ++i)
{
 cout << i << " ";
 }
 return 0;
}
```

While loop:

- The while loop loops through a block of code as long as a specified condition is true:
- **Syntax:**

```
while (condition)
{
  // code block to be executed
}
```

Example:

```
#include <iostream>
using namespace std;
int main()
{
  int i = 0;
  while (i < 5) {
    cout << i << "\n";
    i++;
  }
  return 0;
}
```

- **The Do/While Loop:**

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax:

```
do {
  // code block to be executed
}
while (condition);
```

Example:

The example below uses a do/while loop. the loop will always be executed at least once, even if the condition is false because

- the code block is executed before the condition is tested:

Example:

```
#include <iostream>
using namespace std;

int main()
{
  int i = 0;
  do
 {
    cout << i << "\n";
    i++;
  }
  while (i < 5);
  return 0;
}
```

- **FUNCTION IN C++ :**

The main( ) Function :

ANSI does not specify any return type for the main ( ) function which is the starting point for the execution of a program . The definition of main( ) is :-

```
main()
{
 //main program statements
}
```

This is property valid because the main () in ANSI C does not return any value. In C++, the main () returns a value of type int to the operating system. The functions that have a return value should use the return statement for terminating. The main () function in C++ is therefore defined as follows.

```
int main( )
{
---------------
 ---------------
return(0)
}
```

Since the return type of functions is int by default, the key word int in the main( ) header is optional.

INLINE FUNCTION:

To eliminate the cost of calls to small functions C++ proposes a new feature called inline function. An inline function is a function that is expanded inline when it is invoked .

That is the compiler replaces the function call with the corresponding function code.

The inline functions are defined as follows:-

```
inline function-header
{
 function body;
}
```

Example:

```
inline double cube (double a)
{
  return(a*a*a);
 }
```

The above inline function can be invoked by statements like

- c=cube(3.0);   d=cube(2.5+1.5);
- remember that the inline keyword merely sends a request, not a command to the compiler.
- The compiler may ignore this request if the function definition is too long or too complicated and compile the function as a normal function.
- **FUNCTION OVERLOADING:**

Overloading refers to the use of the same thing for different purposes . C++ also permits overloading functions .This means that we can use the same function name to creates functions that perform a variety of different tasks. This is known as function polymorphism in oops. Using the concepts of function overloading , a family of functions with one function name but with different argument lists in the functions call .The correct function to be invoked is determined by checking the number and type of the arguments but not on the function type.

For example an overloaded add() function handles different types of data as shown below.

```
//Declaration
 int add(int a, int b); //prototype 1
 int add (int a, int b, int c); prototype 2
 double add(double x, double y); //prototype 3
 double add(double p , double q); //prototype4
 //function call
 cout<<add(5,10); //uses prototype 1
cout<<add(5,10,15); //uses prototype 2
cout<<add(12.5,7.5); //uses prototype 3
cout<<add(15,10.0); //uses prototype 4
```