CBSE 12th C++

Break and continue statement

The break statement is used for unconditional termination from the loop
The continue statement is used for taking the control to the beginning of the loop.
For example the following program

```
int I=1;
While(I<=10)
{
cout<<"hello";
I++
break;
}
```

will print hello on the screen just for one time and not for ten times.

While loop

With while loop there are three conditions to be checked
Initialization, where a variable is initialized
Testing, where the condition of the loop is checked
Increment/decrement where we increment or decrement the variable that we have initialized earlier.

Difference in while loop and for loop

In while loop all the three conditions are placed at three different places where as in for loop all the three conditions are placed at the same place .

Difference in while and do-while loop

In while loop the condition is tested in the beginning where as in do-while loop the condition is tested at the end ,therefore the do-while loop can execute atleast once without checking the condition.

Do-while loop example

```
#include
#include
void main()
{
int I=0;
clrscr();
do
{
```

```
cout<<"hello"<<endl;
I++;
}while(I<10);
getch();
}
```

example of a function

```
#include
#include
void main()
{
add();
getch();
}
add()
{
int num1,num2.sum=0;
cout<<"Enter A Number"<<endl;
cin>>num1
cout<<"Enter Another Number"<<endl;
cin>>num2;
sum=num1+num2;
cout<<"The sum of two numbers is "<<sum;
}
```

Here we have divided our job in a separate function add. The functions are reusable i.e. they can be called from a function as many times it is needed.

A function has three different stages
Function declaration or function prototyping
Function calling
Function definition

The most important function in c++ is main(). main() is very nessary for execution of c++ program because compiler first of all searches for the main() function.

There is no boundation for the number of functions in a program
If there is just one function in a program it must be main()
A function can be called any number of times.
A function by default returns an integer value.
Return statement in a function definition returns a value to the calling function
A function which calls another function is called a calling function whereas the other function is referred as called function.
A function ,when called by itself is called recursive function.

An interactive calculator using functions and switch case

```
#include
#include
void main()
{
int ch;
cout<<”1. ADD”<<endl;
cout<<”2 .SUBSTRACION”<<endl;
cout<<”3. MULTIPLICATION”<<endl;
cout<<”4. DIVISION”<<endl;
cout<<”EXIT”<<endl;
cout<<”ENTER YOUR CHOICE”<<endl;
cin>>ch
switch(ch)
{
case 1:
add();
break;
case 2:
sub();
break;
case 3:
mul();
break;
case 4:
div();
break;
case 5:
exit();

default:
cout<<”Invalid Choice”<<endl;
break;
}
getch();
}

add()
{
int num1,num2,sum=0;
cout<<”Enter A Number”<<endl;
cin>>num1
cout<<”Enter Another Number”<<endl;
cin>>num2;
sum=num1+num2;
cout<<”The sum of two numbers is “<<sum;
}


sub()
{
int num1,num2,sub=0
cout<<”Enter A Number”<<endl;
cin>>num1
cout<<”Enter Another Number”<<endl;
cin>>num2;
sub=num1-num2;
cout<<”The diffeerence of two numbers is “<<sub;
}


mul()
{
int num1,num2,mul=0;
cout<<”Enter A Number”<<endl;
cin>>num1
cout<<”Enter Another Number”<<endl;
cin>>num2;
mul=num1*num2;
cout<<”The multiplication of two numbers is “<<mul;
}


div()
{
int num1,num2,div=0;
cout<<”Enter A Number”<<endl;
cin>>num1
cout<<”Enter Another Number”<<endl;
cin>>num2;
div=num1/um2;
cout<<”The division of two numbers is “<<div;
}
```

An example where a function is returning a value

```
#include
#include
void main()
{
int n1,n2,sum=0;
cout<<”Type a number”<<endl;
cin>>n1;
cout<<”type another numer”<<endl;
cin>>n2;
sum=add(n1,n2)
cout<<”The sum of the two numbers is “<<sum<<endl;
getch();
}
add(int x,int y)
{
int z=0;
z=x+y;
return(z);
}
```

Here x y,z are local variable to the function add().When we have called the function add in function main() we have passed two variables n1,n2 to the function add. At the time of definition we have used x and y. the values of n1 is passing to x and the value of n2 is passing to y , so the variable z is having the sum of n1 and n2. with the help of return statement we are returning the value to the calling function i.e. main().

Pointers in c++

Pointer is a special kind of variable which holds the address of another variable. Pointers can be declared for any kind of variable. The pointer should be of the same data type as that of the variable of which it is holding the address. For example for storing the address of an integer variable we must have an integer pointer and so on......

Declaration of a pointer

A pointer can be declared as

```
int *ptr ;
char *p1;
float *fptr;
```

To store the address of a variable we use & (Address Of) operator.

Viz.

```
int I,*ptr;
ptr=&I;
```

a pointer holds the address of the variable. By using *(value at address) operator we can see the value of the variable also.

An example to show all these things

```
#include
#include
void main()
{
int I=10;
int *ptr;
ptr=&I;

// address and value of the variable using pointer
cout<<" The address of the variable I is "<<ptr<<endl;
cout<< "The value of the variable I is "<< *ptr<<endl;
// address and value of the variable simply
cout<<" The address of the variable I is "<<&I <<endl;
cout<< "The value of the variable I is "<< I <<endl;

getch();
}
```

Arrays

Arrays are collection of similar type of elements in a contiguous memory location.
For example int arr[20] will contain 20 elements of integer type in contiguous memory location.
Array could of any data type. char arrays are also referred to as strings.

Example:

```
#include
#include
void main()
{
int arr[10];

// for entering ten numbers in an array
cout<<"enter ten numbers"<<endl;
for(int I=0;I<10;I++)
{
cin>>arr[I];
}
// for displaying those ten numbers entered

for(int I=0;I<10;I++)
{
cout<<arr[i];
}

getch();
}
```

In this example the statement inside first loop are used for entering ten numbers in the array arr.
And the statement inside the second for loop is used for showing the values of the array arr.

```
//Example to enter ten numbers in an array and find out their sum

#include
#include
void main()
{
int arr[10],I,sum=0;
cout<<"Enter Ten numbers"<<endl;
for(int I=0;I<10;I++)
{
cin>>arr[I];
sum=sum+arr[I];
}
cout<<sum;
getch();
}
```

As we discussed earlier that arrays can be of different data types.char arrays are also referred to as strings.
In strings we can take all those entities which are either alphabetical or in some cases alphanumerical such as name,address,telephone number,pincode number etc.

Some functions related to the strings
strlen() this function returns the length of the string viz.

```
#include
#include
#include
void main()
{
int length;
char name[20];
cout<<"type a name"<<endl;
cin>>name;
length=strlen(name);
cout<<"the length of the name is "<<length<<endl;
getch();
}
```

strcpy()
This function is used to copy one string to other.

```
#include
#include
#include
void main()
{
char str[20],str1[20];
cout<<"Enter a string"<<endl;
cin>>str;
strcpy(str1,str);
cout<<" The copied string is "<< str1<<endl;
getch();
}
```

strcat()

This function is used to add one string at the end of the other
the syntax is

strcat(target, source)

Here source and target are two strings. If the source string has world and target string has hello then after concatenation the target string will contain Hello world.

strcmp()

This function is used to compare two strings. The function returns an integer value . It may be a zero, a negative value or a positive value.

Syntax:
strcmp(str1,str2)

Some other functions strrev(),strncat(),strcmpi() etc.

object
The fundamental idea behind an object-oriented language is to combine into a single unit both data and the functions that operate on the data. Such a unit is called an object.
An object’s function provides the only way to access the data. Thus the data is hidden and is safe from accidental alteration.
Classes:

As we know that objects contain data and code to manipulate that data. The entire set of data and code of an object can be made a user defined datatype with the help of a class. Once a class has been defined we can create any number of objects belonging to that class.

A class is thus a collection of objects of similar type. For example mango,apple and orange are members of the class fruit.

If fruit is a class then the statement
Fruit banana will create an object banana belonging to class fruit.

Inheritence
Inheritence is the process of creating a new class, called the derived class, from the existing class, called the base class.

In some other words we can say that inheritence is the process by which objects of one class can acquire the properties of the objects of another class.

In OOP the concept of inheritence provides the idea of reusability. This means that we can add additional freaures to an existing class without modifying it. This is possible by deriving a new class from the existing class. The new class will have the combined features of both the classes.

Polymorphism
The word polymorphism is derived from two latin words poly(many) and morphos (forms). The concept of using operators or functions in different ways, depending on what they are operating on, is called polymorphism.

Data abstraction and Encapsulation
The wrapping up of data and function into a single unit(called class) is known as encapsulation.
Data encapsulation is the most striking feature of a class. The data is not accessible to the outside world and only those function which are wrapped in the class can access it.
Abstraction refers to the act of representing essential features without including the background details or explanation. Since the classes uses the concept of data abstraction, they are known as abstract data types.

An example to illustrate class concept

```
#include
#include
class add
{
int n1,n2,sum;
public:
accept();
sum();

display();
};
add::accept()
{
cout<<”Type a number”<<endl;
cin>>n1;
cout<<”Enter another number”<<endl;
cin>>n2;
}
add:: sum()
{
sum=n1+n2;
}
add::display()
{
cout<<” The sum of the two numbers”<<sum;
}

void main()
{
add a1;
a1.accept();
a1.sum();
a1.display();
getch();
}
```

An example to illustrate class concept

```
#include
#include
class add
{
int n1,n2,sum;
public:
accept()
{
cout<<”Type a number”<<endl;
cin>>n1;
cout<<”Enter another number”<<endl;
cin>>n2;
}
sum()
{
sum=n1+n2;
}
display()
{
cout<<" The sum of the two numbers"<<sum;
}
};// END OF THE CLASS DECLARATION
void main()
{
add a1;
a1.accept();
a1.sum();
a1.display();
getch();
}
```

Here we have defined a class with the name add. Here we have two integer variables which are to be accepted from the user. We have an accept function for the same. To add these two numbers we have a function sum . and to display the sum we have a function display. In this example we have declared all the function inside the class declaration. We can define these functions outside the class declaration also.
In main function we have created an object (a1) of class add. As we know that the data of the class can only be accessed with the help of objects. that's why we are calling all the functions of the class add using object a1.

```
#include
#include
class base
{
int num1;
public:
accept()
{
cout<<"Enter A number"<<endl;
cin>>num;
}
display()
{
cout<<"the value you have entered is "<<num<<endl;
}
};
class derived : public base
{
}

class integer
```

```
{
int I;
public:
void getdata()
{
cout<<”Enter an integer value”<<endl;
cin>>I:
}
void setdata( int j)
{
I=j;
}
integer()
{
I=0;
}
integer(int k)
{
I=k;
}
void display()
{
cout<<”value of I is “<<i<<endl;
}
};
void main()
{
integer i1(100),i2,i3;
i1.display();
i2.setdata(200);
i2.display();
i3.getdata();
i3.display();
}
```

this program shows three methods ti assign values to the class data using objects.

Constructors are special functions with the same name as that of the class. They are automatically called when an instance of the class is created. Constructors are not same as that of functions because we never use dot(.) operator to call a constructor.

```
Class example
{
public:

example()
{
cout<<”Inside the constructor”<<endl;
}
~example()
{
```

```
cout<<"inside the destructor"<<endl;
}
};

void main()
{
example e;
}


</endl;
</endl;
</i<<endl;
</endl;
</num<<endl;
</endl;
</sum;
</endl;
</endl;
</sum;
</endl;
</endl;
</endl;
</endl;
</length<<endl;
</endl;
</sum;
</endl;
</arr[i];
</endl;
</endl;
</endl;
</endl;
</ptr<<endl;
</sum<<endl;
</endl;
</endl;
</div;
</endl;
</endl;
</mul;
</endl;
</endl;
</sub;
</endl;
</endl;
</sum;
</endl;
</endl;
</endl;
</endl;
</endl;
</endl;
</endl;
</endl;
```

```
</endl;
</sum;
</endl;
</endl;
</endl;
```