OPERATORS IN C++ :

C++ has a rich set of operators. All C operators are valid in C++ also. In addition. C++ introduces some new operators.

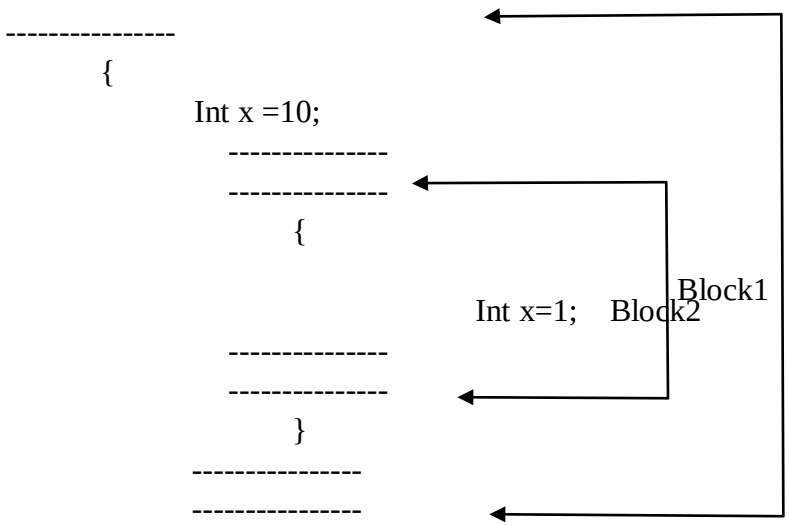
<<	insertion operator
>>	extraction operator
: :	scope resolution operator
: :*	pointer to member declarator
*	pointer to member operator
.*	pointer to member operator
Delete	memory release operator
Endl	line feed operator
New	memory allocation operator
Setw	field width operator

SCOPE RESOLUTION OPERATOR:

Like C,C++ is also a block-structured language. Block -structured language. Blocks and scopes can be used in constructing programs. We know same variables can be declared in different blocks because the variables declared in blocks are local to that function.

Blocks in C++ are often nested.

Example:



}

Block2 contained in block l .Note that declaration in an inner block hides a declaration of the same variable in an outer block and therefore each declaration of x causes it to refer to a different data object .

With in the inner block the variable x will refer to the data object declared there in.

In C,the global version of a variable can't be accessed from with in the inner block. C++ resolves this problem by introducing a new operator :: called the scope resolution operator .This can be used to uncover a hidden variable.

Syntax: **: : variable –name;**

Example:

```
#include
    <iostrcam.h
    > int m=10;
    main()
  {
 int m=20;
    {  int
    k=m
    ;  int
    m=3
    0;
    cout<<”we are in inner
    block”;
    cout<<"k="<<k<<endl;
    cout<<"m="<<m<<endl;
    cout<<":: m="<<:: m<<endl;
  }
cout<<”\n we are in outer block \n”;
   cout<<"m="<<m<<endl;
   cout<<":: m="<<:: m<<endl;
```

}

Memory Management Operator

C uses malloc and calloc functions to allocate memory dynamically at run time . Similarly it uses the functions Free( ) to free dynamically allocated memory. We use dynamic allocation techniques when it is not known in advance how much of memory space as needed .

C++ also support those functions it also defines two unary operators new and delete that perform the task of allocating and freeing the memory in a better and easier way.

The new operator can be used to create objects of any type. **Syntax: pointer-**

variable =new datatype;

Example:p=new int; q=new int;

Where p is a pointer of type int and q is a pointer of type float.

int *p=new int;

float *p=new float;

Subsequently, the statements

*p=25;
*q=7.5;

Assign 25 to the newly created int object and 7.5 to the float object.We can also initialize the memory using the new operator.

Syntax: **int *p=ne\v int(25);**
float *q =new float(7.5);

new can be used to create a memory space for any data type including user defined such as arrays,structures,and classes .The general form for a one-dimensional array is:

pointer-variable =new data types [size];

creates a memory space for an array of 10 integers.

If a data object is no longer needed, it is destroyed to release the memory space for reuse.

Syntax: delete pointer-variable;

Example:
delete p;
delete q;

If we want to free a dynamically allocated array ,we must use the following form of delete.

delete [size] pointer-variable;

or

delete [ ] pointer variable;

MANIPULATERS:

Manipulators are operator that are used to format the data display. The most commonly manipulators are endl and setw.

The endl manipulator, when used in an output statement, causes a line feed to be insert.(just like \n)

Example:

```
cout<<"m="<<m<<endl; cout<<"n="<<n<<endl;
cout<<"p="<<p<<endl;
```

If we assume the values of the variables as 2597,14 and 175 respectively m=2597; n=14; p=175

It was want to print all nos in right justified way use setw which specify a common field width for all the nos.

Example:

```
cout<<setw(5)<<sum<<endl;
cout<<setw(10)<<"basic"<<setw(10<<basic<<endl;
Cout<<setw(10)<<"allowance"<<setw(10<<allowance<<endl;
cout<<setw(10)<<"total="<<setw(10)<<total;
```