21UCU403 - OBJECT ORIENTED PROGRAMMING

UNIT - 2

Pointer and I/O and File Management

Section - 1

1. Infer on Inline functions.

Answer:

- C++ inline function is a powerful concept that is commonly used with classes.
- If a function is inline, the compiler places a copy of the code of that function at each point where the function is called at compile time.
- Any change to an inline function could require all clients of the function to be recompiled because the compiler would need to replace all the code once again otherwise it will continue with old functionality.
- To inline a function, place the keyword inline before the function name and define the function before any calls are made to the function.
- The compiler can ignore the inline qualifier in case a defined function is more than a line.
- A function definition in a class definition is an inline function definition, even without the use of the inline specifier.

Example:

```
#include <iostream>
using namespace std;
inline int Max(int x, int y) {
   return (x > y)? x : y;
}
// Main function for the program
int main() {
   cout << "Max (20,10): " << Max(20,10) << endl;
   cout << "Max (0,200): " << Max(0,200) << endl;
   cout << "Max (100,1010): " << Max(100,1010) << endl;
   return 0;
}
```

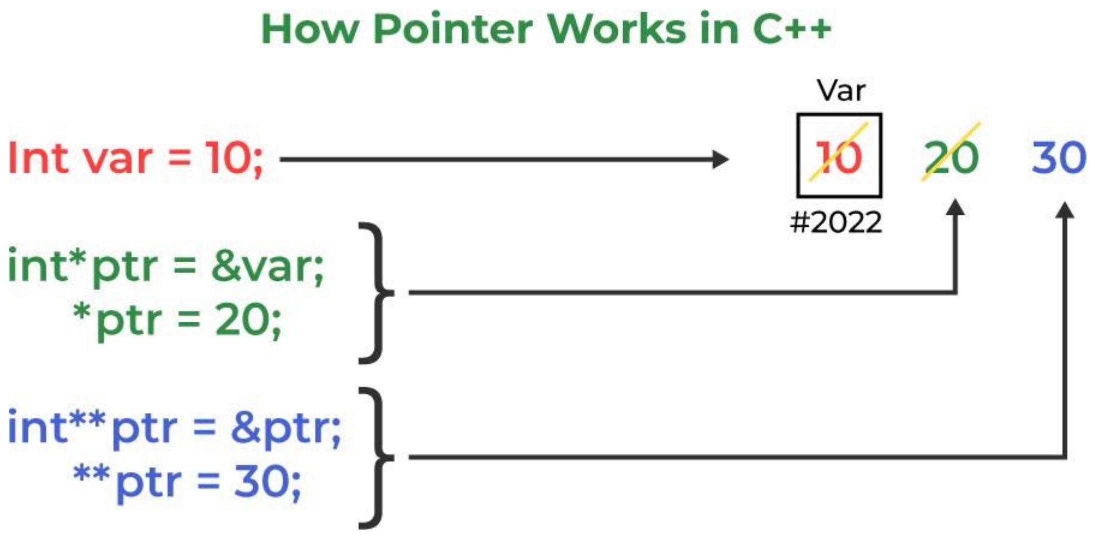
Output:

```
Max (20,10): 20
Max (0,200): 200
Max (100,1010): 1010
```

2. Summarize on Pointers in C++.

Answer:

- Pointers are symbolic representations of addresses.
- They enable programs to simulate call-by-reference as well as to create and manipulate dynamic data structures.
- Iterating over elements in arrays or other data structures is one of the main uses of pointers.
- The address of the variable you're working with is assigned to the pointer variable that points to the same data type (such as an int or string).



Syntax:

```
datatype *var_name;
int *ptr;   // ptr can point to an address which holds int data
```

Example:

```
#include <bits/stdc++.h>
using namespace std;
void geeks()
{
   int var = 20;
```

```
    // declare pointer variable
    int* ptr;
    // note that data type of ptr and var must be same
    ptr = &var;
    // assign the address of a variable to a pointer
    cout << "Value at ptr = " << ptr << "\n";
    cout << "Value at var = " << var << "\n";
    cout << "Value at *ptr = " << *ptr << "\n";
}
// Driver program
int main()
{
  geeks();
  return 0;
}
```

Output;

```
Value at ptr = 0x7ffe454c08cc
Value at var = 20
Value at *ptr = 20
```

3. Discuss the “this” pointer.

Answer:

In C++ programming, “this” is a keyword that refers to the current instance of the class. There can be 3 main uses of this keyword in C++.

- It can be used to pass the current object as a parameter to another method.
- It can be used to refer to the current class instance variable.
- It can be used to declare indexers.

Example:

```
#include <iostream>
using namespace std;
class Employee {
   public:
       int id; //data member (also instance variable)
       string name; //data member(also instance variable)
       float salary;
       Employee(int id, string name, float salary)
        {
             this->id = id;
            this->name = name;
            this->salary = salary;
        }
       void display()
        {
            cout<<id<<"  "<<name<<"  "<<salary<<endl;
        }
};
int main(void) {
    Employee e1 =Employee(101, "Sonoo", 890000); //creating an object of
Employee
    Employee e2=Employee(102, "Nakul", 59000); //creating an object of
Employee
    e1.display();
    e2.display();
    return 0;
}
```

Output:

```
101  Sonoo  890000
102  Nakul  59000
```

4. What is virtual and pure virtual functions.

Answer:

Virtual function

- A virtual function is a member function of base class which can be redefined by derived class.
- Classes having virtual functions are not abstract.
- Definition is given in base class.

- Base class having virtual function can be instantiated i.e. its object can be made.
- If derived classes do not redefine the virtual function of base class, then it does not affect compilation.
- All derived classes may or may not redefine the virtual function of the base class.

Syntax:

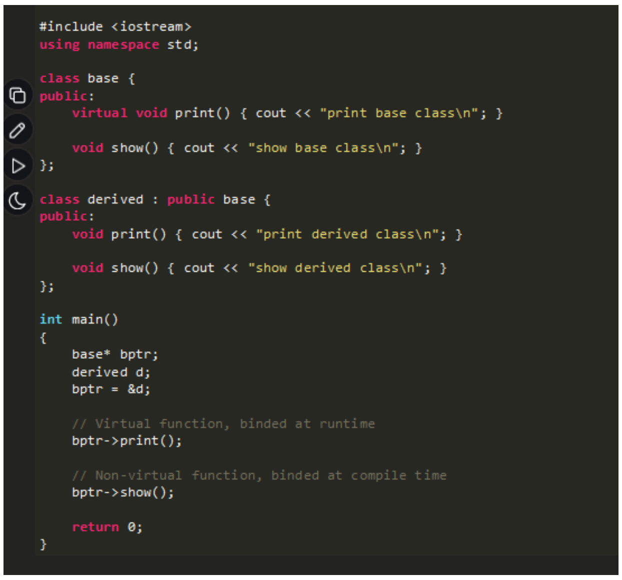
```
virtual<func_type><func_name>()
{
        // code
}
```

Pure virtual function

- A pure virtual function is a member function of base class whose only declaration is provided in base class and should be defined in derived class otherwise derived class also becomes abstract.
- Base class containing pure virtual function becomes abstract. No definition is given in base class.
- Base class having pure virtual function becomes abstract i.e. it cannot be instantiated.
- If the derived class does not redefine the virtual function of base class, then no compilation error but derived class also becomes abstract just like the base class.
- All derived classes must redefine the pure virtual function of base class otherwise derived class also becomes abstract just like base class.

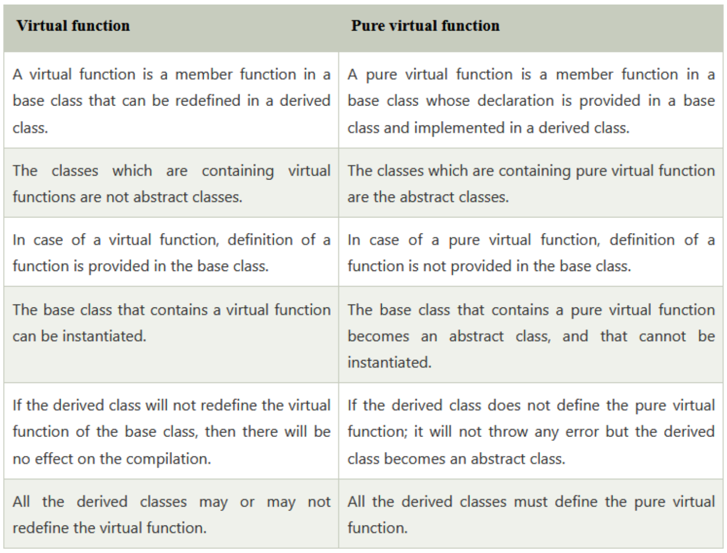
Syntax:

```
virtual<func_type><func_name>()
        = 0;
```



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

class base {
public:
    virtual void print() { cout << "print base class\n"; }

    void show() { cout << "show base class\n"; }
};

class derived : public base {
public:
    void print() { cout << "print derived class\n"; }

    void show() { cout << "show derived class\n"; }
};

int main()
{
    base* bptr;
    derived d;
    bptr = &d;

    // Virtual function, binded at runtime
    bptr->print();

    // Non-virtual function, binded at compile time
    bptr->show();

    return 0;
}
```

Difference between virtual function and pure virtual function



Virtual function	Pure virtual function
A virtual function is a member function in a base class that can be redefined in a derived class.	A pure virtual function is a member function in a base class whose declaration is provided in a base class and implemented in a derived class.
The classes which are containing virtual functions are not abstract classes.	The classes which are containing pure virtual function are the abstract classes.
In case of a virtual function, definition of a function is provided in the base class.	In case of a pure virtual function, definition of a function is not provided in the base class.
The base class that contains a virtual function can be instantiated.	The base class that contains a pure virtual function becomes an abstract class, and that cannot be instantiated.
If the derived class will not redefine the virtual function of the base class, then there will be no effect on the compilation.	If the derived class does not define the pure virtual function; it will not throw any error but the derived class becomes an abstract class.
All the derived classes may or may not redefine the virtual function.	All the derived classes must define the pure virtual function.

5. Discuss on cin and cout objects.

Answer:

- C++ comes with libraries that provide us with many ways for performing input and output.

- In C++ input and output are performed in the form of a sequence of bytes or more commonly known as streams.
- Input Stream: If the direction of flow of bytes is from the device(for example, Keyboard) to the main memory then this process is called input.
- Output Stream: If the direction of flow of bytes is opposite, i.e. from main memory to device(display screen) then this process is called output.

Example:

```
#include <iostream>
using namespace std;
int main()
{
        int age;
        cout << "Enter your age:";
        cin >> age;
        cout << "\nYour age is: " << age;
        return 0;
}
```

Output:

```
Enter your age: 18
Your age is: 18
```

6. What is C++ stream classes.

Answer:

- Stream in C++ means a stream of characters that gets transferred between the program thread and input or output.
- There are a number of C++ stream classes eligible and defined which are related to the files and streams for providing input-output operations.
- All the classes and structures maintaining the file and folders with hierarchies are defined within the file with iostream.h standard library.
- Classes associated with the C++ stream include ios class, istream class, and ostream class.

- Class ios is indirectly inherited from the base class involving iostream class using istream class and ostream class which is declared virtually.

istream Class

istream being a part of the ios class which is responsible for tackling all the input stream present within the stream. It provides all the necessary and important functions with the number of functions for handling all the strings, chars, and objects within the istream class which comprises all these functions such as get, read, put, etc.

ostream Class

This class as part of the ios class is also considered as a base class that is responsible for handling output stream and provides all the necessary functions for handling chars, strings, and objects such as put, write, etc.

iostream Class

iostream class is the next hierarchy for the ios class which is essential for input stream as well as output stream because istream class and ostream class gets inherited into the main base class. As the name suggests it provides functionality to tackle the objects, strings, and chars which includes inbuild functions of put, puts, get, etc.

Section - 2

1. Discuss on unformatted I/O in C++.

Answer:

Overloaded Operators >> and <<:

We have used the objects cin and cout for the input and output of data of various types. The >> operator is overloaded in the istream class and << is overloaded in the ostream class.

Syntax:
cin >> var1 >> var2 >> …. >> var_n;

- Here, var1, var2, ……, var_n are the variable names that are declared already.
- The input data must be separated by white space characters and the data type of user input must be similar to the data types of the variables which are declared in the program.
- The operator >> reads the data character by character and assigns it to the indicated location.
- Reading of variables terminates when white space occurs or character type occurs that does not match the destination type.

put() and get() functions:

- The classes istream and ostream define two member functions get() and put() are used to handle the single character input/output operations.
- There are two types of get() functions.
- We can use both get(char*) and get(void) prototype to fetch a character including the blank space, tab and the newline character.
- The get(char*) version assigns the input character to its argument and the get(void) version returns the input character.

Syntax:

```
char data;
// get() return the character value and assign to data variable
data = cin.get();
// Display the value stored in data variable
cout.put(data);
```

2. Describe about formatted I/O in C++.

Answer:

C++ helps you to format the I/O operations like determining the number of digits to be displayed after the decimal point, specifying number base etc. C++ supports a number of features that could be used for formatting the output. These features include:

- ios class functions and flags

- Manipulators

Formatting using the ios members

- The stream has the format flags that control the way of formatting. Using this setf function, we can set the flags, which allow us to display a value in a particular format.
- The ios class declares a bitmask enumeration called fmtflags in which the values(showbase, showpoint, oct, hex etc) are defined.
- These values are used to set or clear the format flags.

Function	Task
width()	It specifies the required field size for displaying an output value.
precision()	It specifies the number of digits to be displayed after the decimal point of a float value.
fill()	It specifies a character that is used to fill the unused portion of a field.
setf()	It specifies format flags that can control the form of the output display.
unsetf()	It is used to clear the flags specified.

Formatting using Manipulators

- The second way you can alter the format parameters of a stream is through the use of special functions called manipulators that can be included in an I/O expression.

Manipulators	Task
setw()	It is used to specify the required field size for displaying an output value.
setprecision()	It is used to specify the number of digits to be displayed after the decimal point of a float value.

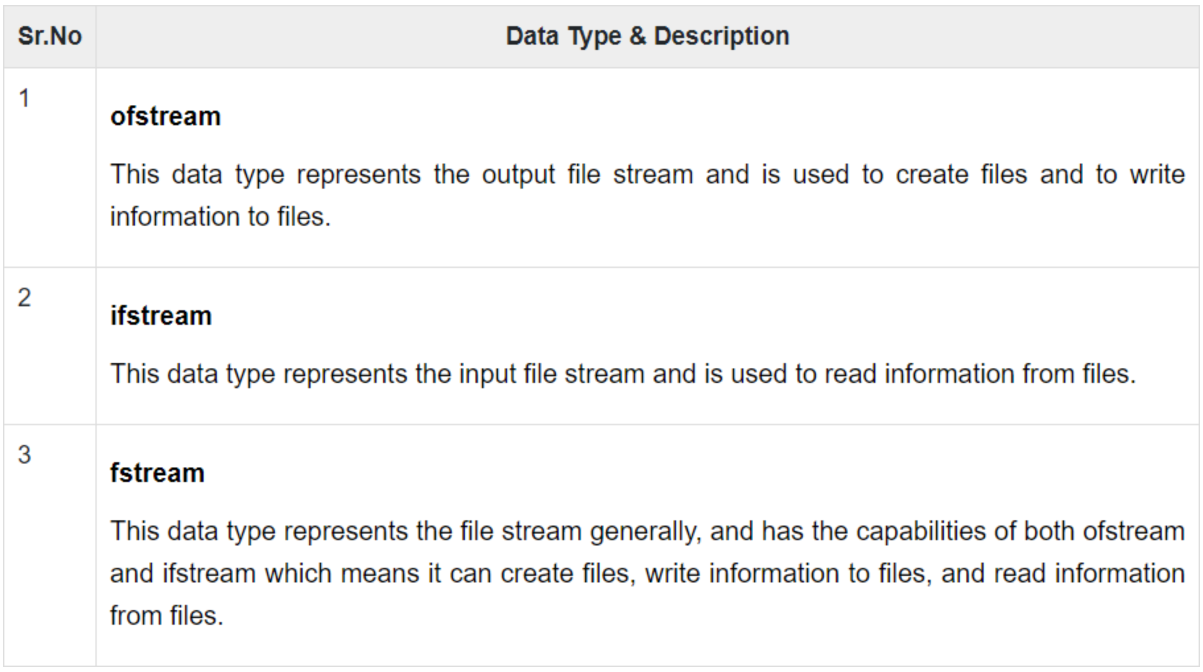
setfill()	It is used to specify a character that is used to fill the unused portion of a field.
setiosflags()	It is used to specify format flags that can control the form of the output display.
resetiosflags()	It is used to clear the flags specified.

3. What is File stream? Explain.

Answer:

In C++ programming we are using the iostream standard library, it provides cin and cout methods for reading from input and writing to output respectively.

To read and write from a file we are using the standard C++ library called fstream. Let us see the data types define in fstream library is:



Sr.No	Data Type & Description
1	**ofstream** This data type represents the output file stream and is used to create files and to write information to files.
2	**ifstream** This data type represents the input file stream and is used to read information from files.
3	**fstream** This data type represents the file stream generally, and has the capabilities of both ofstream and ifstream which means it can create files, write information to files, and read information from files.

C++ FileStream example: writing to a file

Example of writing to a text file testout.txt using C++ FileStream programming.

```
#include <iostream>
#include <fstream>
```

```
using namespace std;
int main () {
  ofstream filestream("testout.txt");
  if (filestream.is_open())
  {
filestream << "Welcome to javaTpoint.\n";
    filestream << "C++ Tutorial.\n";
    filestream.close();
  }
  else cout <<"File opening is fail.";
  return 0;
}
```

Output:

```
The content of a text file testout.txt is set with the data:
Welcome to javaTpoint.
C++ Tutorial.
```

C++ FileStream example: reading from a file

Example of reading from a text file testout.txt using C++ FileStream programming.

```
#include <iostream>
#include <fstream>
using namespace std;
int main () {
  string srg;
  ifstream filestream("testout.txt");
  if (filestream.is_open())
  {
    while ( getline (filestream,srg) )
    {
      cout << srg <<endl;
    }
    filestream.close();
  }
  else {
      cout << "File opening is fail."<<endl;
    }
  return 0;
}
```

Output:

```
Welcome to javaTpoint.
C++ Tutorial.
```

4. Explain on File Management.

Answer:

File Handling In C++

- Files are used to store data in a storage device permanently.
- File handling provides a mechanism to store the output of a program in a file and to perform various operations on it.
- A stream is an abstraction that represents a device on which operations of input and output are performed.
- A stream can be represented as a source or destination of characters of indefinite length depending on its usage.
- In C++ we have a set of file handling methods.
- These include ifstream, ofstream, and fstream.
- These classes are derived from fstrembase and from the corresponding iostream class.
- These classes, designed to manage the disk files, are declared in fstream and therefore we must include fstream and therefore we must include this file in any program that uses files.

In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream.

ofstream: This Stream class signifies the output file stream and is applied to create files for writing information to files
ifstream: This Stream class signifies the input file stream and is applied for reading information from files
fstream: This Stream class can be used for both read and write from/to files.

All the above three classes are derived from fstreambase and from the corresponding iostream class and they are designed specifically to manage disk files.

C++ provides us with the following operations in File Handling:

- ★ Creating a file: open()
- ★ Reading data: read()
- ★ Writing new data: write()
- ★ Closing a file: close()

5. Discuss in detail on File Management functions.

Answer:

Opening a File

- To read or enter data to a file, we need to open it first.
- This can be performed with the help of ‘ifstream’ for reading and ‘fstream’ or ‘ofstream’ for writing or appending to the file.
- All these three objects have an open() function pre-built in them.

Syntax:

open(FileName , Mode);

Here:
FileName – It denotes the name of a file which has to be opened.
Mode – There are different modes to open a file.

Writing to File

- We will use fstream or ofstream objects to write data into the file and to do so; we will use stream insertion operator (<<) along with the text enclosed within the double-quotes.
- With the help of open() function, we will create a new file named ‘FileName’ and then we will set the mode to ‘ios::out’ as we have to write the data to file.

Syntax:

FileName<<"Insert the text here";

Reading from file

- Getting the data from the file is an essential thing to perform because without getting the data, we cannot perform any task.
- We can perform the reading of data from a file with the CIN to get data from the user, but then we use CIN to take inputs from the user’s standard console.
- Here we will use fstream or ifstream.

Syntax:

FileName>>Variable;

Closing a file in C++

- Closing a file is a good practice, and it is a must to close the file.
- Whenever the C++ program comes to an end, it clears the allocated memory, and it closes the file. We can perform the task with the help of close() function.

Syntax:

FileName.close();

6. How does C++ File stream classes?

Answer:

- ios class is the topmost class in the stream classes hierarchy. It is the base class for istream, ostream, and streambuf class.
- istream and ostream serve the base classes for iostream class. The class istream is used for input and ostream for the output.
- Class ios is indirectly inherited to iostream class using istream and ostream.

Facilities provided by these stream classes.

The ios class: The ios class is responsible for providing all input and output facilities to all other stream classes.

The istream class: This class is responsible for handling input streams. It provides a number of functions for handling chars, strings and objects such as get, getline, read, ignore, putback etc..

Example:

```
#include <iostream>
using namespace std;
int main()
{
    char x;
    // used to scan a single char
    cin.get(x);
    cout << x;
}
```

Output:

```
g
g
```

The ostream class: This class is responsible for handling output stream. It provides a number of functions for handling chars, strings and objects such as write, put etc..

Example:

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

int main()
{
    char x;
    // used to scan a single char
    cin.get(x);
    // used to put a single char onto the screen.
    cout.put(x);
}
```

Output:

```
h
h
```

The iostream: This class is responsible for handling both input and output stream as both istream class and ostream class is inherited into it. It provides functions of both istream class and ostream class for handling chars, strings and objects such as get, getline, read, ignore, putback, put, write etc..

Example:

```
#include <iostream>
using namespace std;
int main()
{
    // this function display
    // ncount character from array
    cout.write("hellothere", 5);
}
```

Output:

```
hello
```

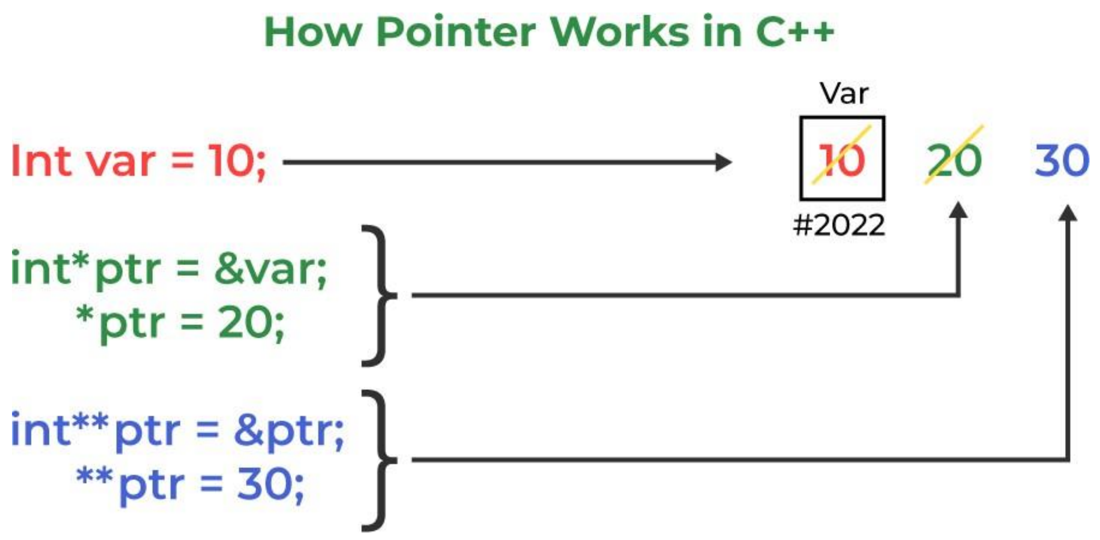
Section - 3

1. Summarize on Pointers and Objects.

Answer:

- Pointers are symbolic representations of addresses.
- They enable programs to simulate call-by-reference as well as to create and manipulate dynamic data structures.

- Iterating over elements in arrays or other data structures is one of the main uses of pointers.
- The address of the variable you’re working with is assigned to the pointer variable that points to the same data type (such as an int or string).



Syntax:

```
datatype *var_name;
int *ptr;   // ptr can point to an address which holds int data
```

Example:

```
#include <bits/stdc++.h>
using namespace std;
void geeks()
{
    int var = 20;
    // declare pointer variable
    int* ptr;
    // note that data type of ptr and var must be same
    ptr = &var;
    // assign the address of a variable to a pointer
    cout << "Value at ptr = " << ptr << "\n";
    cout << "Value at var = " << var << "\n";
    cout << "Value at *ptr = " << *ptr << "\n";
}
// Driver program
int main()
{
  geeks();
  return 0;
}
```

Output;

```
Value at ptr = 0x7ffe454c08cc
Value at var = 20
Value at *ptr = 20
```

C++ allows you to have pointers to objects. The pointers pointing to objects are referred to as "object pointers”. Object pointers, like other pointers, are declared by placing in front of the object pointer's name.

It takes the following general form:

class-name ∗object-pointer;

where class-name is the name of an already-defined class, and object-pointer is the pointer to an object of this class type. For example, to declare optr as an object pointer of the Sample class type, we shall write

Sample ∗optr;

where Sample is a predefined class. When using an object pointer to access members of a class, the arrow operator (->) is used instead of the dot operator.

2. Discuss in Detail on Unformatted and formatted I/O.

Answer:

Unformatted I/O:

Overloaded Operators >> and <<:

We have used the objects cin and cout for the input and output of data of various types. The >> operator is overloaded in the istream class and << is overloaded in the ostream class.

Syntax:

```
cin >> var1 >> var2 >> …. >> var_n;
```

- Here, var1, var2, ……, var_n are the variable names that are declared already.
- The input data must be separated by white space characters and the data type of user input must be similar to the data types of the variables which are declared in the program.
- The operator >> reads the data character by character and assigns it to the indicated location.
- Reading of variables terminates when white space occurs or character type occurs that does not match the destination type.

put() and get() functions:

- The classes istream and ostream define two member functions get() and put() are used to handle the single character input/output operations.
- There are two types of get() functions.
- We can use both get(char*) and get(void) prototype to fetch a character including the blank space, tab and the newline character.
- The get(char*) version assigns the input character to its argument and the get(void) version returns the input character.

Syntax:

```
char data;
// get() return the character value and assign to data variable
data = cin.get();
// Display the value stored in data variable
cout.put(data);
```

Formatted I/O:

C++ helps you to format the I/O operations like determining the number of digits to be displayed after the decimal point, specifying number base etc. C++ supports a number of features that could be used for formatting the output. These features include:

- ios class functions and flags
- Manipulators

Formatting using the ios members

- The stream has the format flags that control the way of formatting. Using this setf function, we can set the flags, which allow us to display a value in a particular format.
- The ios class declares a bitmask enumeration called fmtflags in which the values(showbase, showpoint, oct, hex etc) are defined.
- These values are used to set or clear the format flags.

Function	Task
width()	It specifies the required field size for displaying an output value.
precision()	It specifies the number of digits to be displayed after the decimal point of a float value.
fill()	It specifies a character that is used to fill the unused portion of a field.
setf()	It specifies format flags that can control the form of the output display.
unsetf()	It is used to clear the flags specified.

Formatting using Manipulators

- The second way you can alter the format parameters of a stream is through the use of special functions called manipulators that can be included in an I/O expression.

Manipulators	Task
setw()	It is used to specify the required field size for displaying an output value.
setprecision()	It is used to specify the number of digits to be displayed after the decimal point of a float value.
setfill()	It is used to specify a character that is used to fill the unused portion of a field.
setiosflags()	It is used to specify format flags that can control the form of the output display.
resetiosflags()	It is used to clear the flags specified.

3. What are File management functions? Explain.

Answer:

File Handling In C++

- Files are used to store data in a storage device permanently.
- File handling provides a mechanism to store the output of a program in a file and to perform various operations on it.
- A stream is an abstraction that represents a device on which operations of input and output are performed.
- A stream can be represented as a source or destination of characters of indefinite length depending on its usage.
- In C++ we have a set of file handling methods.
- These include ifstream, ofstream, and fstream.
- These classes are derived from fstrembase and from the corresponding iostream class.
- These classes, designed to manage the disk files, are declared in fstream and therefore we must include fstream and therefore we must include this file in any program that uses files.

In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream.

ofstream: This Stream class signifies the output file stream and is applied to create files for writing information to files
ifstream: This Stream class signifies the input file stream and is applied for reading information from files
fstream: This Stream class can be used for both read and write from/to files.
All the above three classes are derived from fstreambase and from the corresponding iostream class and they are designed specifically to manage disk files.

C++ provides us with the following operations in File Handling:

- ★ Creating a file: open()
- ★ Reading data: read()
- ★ Writing new data: write()
- ★ Closing a file: close()

Opening a File

- To read or enter data to a file, we need to open it first.
- This can be performed with the help of 'ifstream' for reading and 'fstream' or 'ofstream' for writing or appending to the file.
- All these three objects have an open() function pre-built in them.

Syntax:

open(FileName , Mode);

Here:
FileName – It denotes the name of a file which has to be opened.
Mode – There are different modes to open a file.

Writing to File

- We will use fstream or ofstream objects to write data into the file and to do so; we will use stream insertion operator (<<) along with the text enclosed within the double-quotes.
- With the help of open() function, we will create a new file named 'FileName' and then we will set the mode to 'ios::out' as we have to write the data to file.

Syntax:

FileName<<"Insert the text here";

Reading from file

- Getting the data from the file is an essential thing to perform because without getting the data, we cannot perform any task.
- We can perform the reading of data from a file with the CIN to get data from the user, but then we use CIN to take inputs from the user's standard console.
- Here we will use fstream or ifstream.

Syntax:

FileName>>Variable;

Closing a file in C++

- Closing a file is a good practice, and it is a must to close the file.
- Whenever the C++ program comes to an end, it clears the allocated memory, and it closes the file. We can perform the task with the help of close() function.

Syntax:

FileName.close();