

Practical-1

Aim: WAP to create a Message class with a constructor that takes a single string with a default value. Create a private member string, and in the constructor simply assign the argument string to your internal string. Create two overloaded member functions called print(): one that takes no arguments and simply prints the message stored in the object, and one that takes a string argument, which it prints in addition to the internal message.

Program:

```
#include<iostream>
#include<string.h>
using namespace std;

class Message
{
    private:
        char n[100];

    public:
        Message(char str[])
        {
            strcpy(this->n,str);
        }

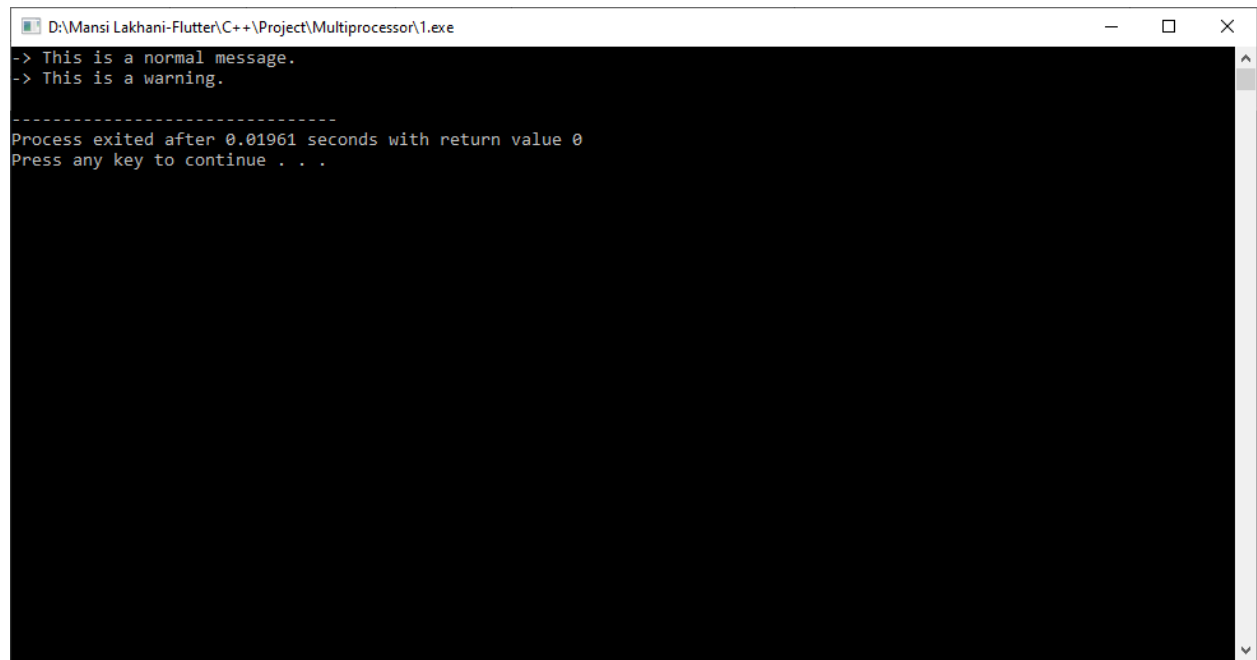
        void print()
        {
            cout << this->n << endl;
        }

        void print(char ch[])
        {
            cout << ch << endl;
        }
};
```

```
int main()
{
    Message m1("-> This is a normal message.");
    m1.print();
    m1.print("-> This is a warning.");

    return 0;
}
```

Output:



A screenshot of a Windows command prompt window. The title bar at the top reads "D:\Mansi Lakhani-Flutter\C++\Project\Multiprocessor\1.exe" and includes standard minimize, maximize, and close buttons. The command prompt area has a black background with white text. The output shows two lines of messages: "-> This is a normal message." and "-> This is a warning.", followed by a separator line of dashes. Below this, it states "Process exited after 0.01961 seconds with return value 0" and "Press any key to continue . . .". A vertical scrollbar is visible on the right side of the window.

```
D:\Mansi Lakhani-Flutter\C++\Project\Multiprocessor\1.exe
-> This is a normal message.
-> This is a warning.
-----
Process exited after 0.01961 seconds with return value 0
Press any key to continue . . .
```

Practical-2

Aim: WAP which illustrate the use of Method Overriding concept.

Program:

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

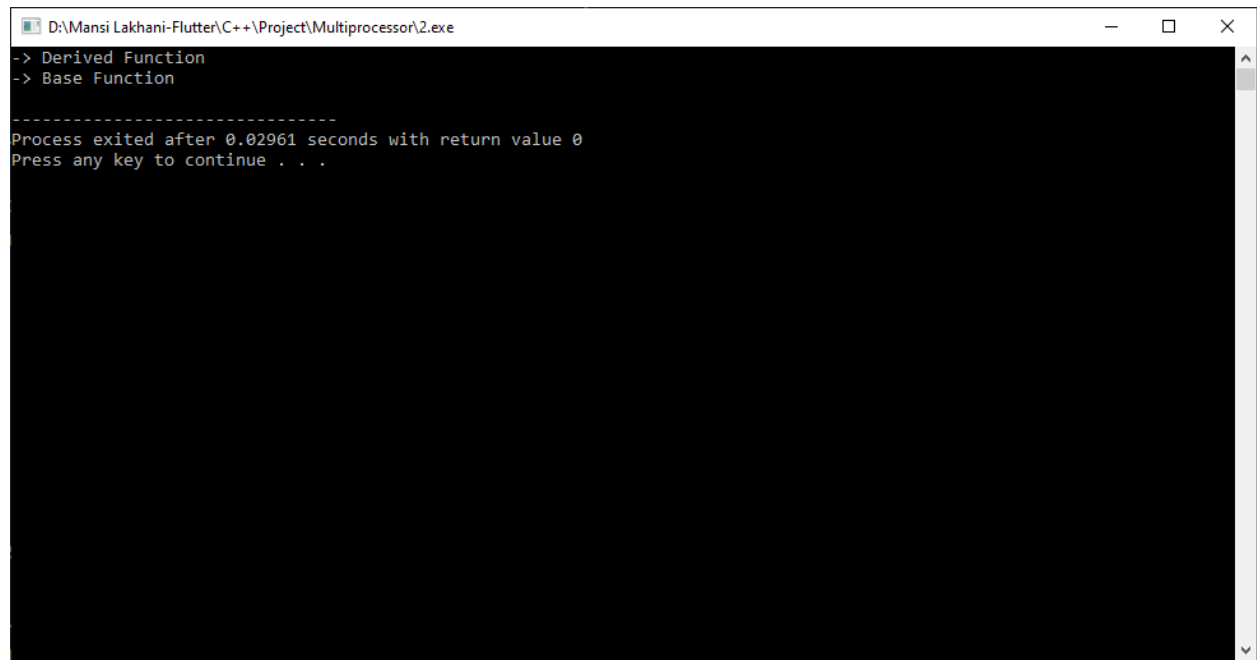
class Base
{
    public:
        void print()
        {
            cout << "-> Base Function"<<endl;
        }
};

class Derived : public Base
{
    public:
        void print()
        {
            cout << "-> Derived Function"<<endl;
            Base::print();
        }
};

int main()
{
    Derived d1;

    d1.print();
    return 0;
}
```

Output:



```
D:\Mansi Lakhani-Flutter\C++\Project\Multiprocessor\2.exe
-> Derived Function
-> Base Function
-----
Process exited after 0.02961 seconds with return value 0
Press any key to continue . . .
```