**Experiment No: 3 Date:**

**Aim: To study concept of constructors and Destructors in C++ programming**

**Theory:**

**Constructors :**

Constructors are special member functions in C++ that are automatically called when an object of a class is created. Their primary purpose is to initialize the object's data members and perform any necessary setup. Key points about constructors:

Constructors have the same name as the class.

They do not have a return type (not even **void**).

You can have multiple constructors with different parameter lists, enabling object initialization in various ways.

If you don't provide a constructor, C++ provides a default constructor that initializes members to default values (e.g., 0 for numbers, an empty string for strings).

Example:

**Destructors :**

Destructors are special member functions used to clean up resources and perform necessary cleanup when an object goes out of scope or is explicitly deleted. Key points about destructors:

Destructors have the same name as the class but preceded by a tilde **~**.

They do not take any parameters.

You usually define a destructor when your class manages resources like memory or file handles.

If you don't provide a destructor, C++ provides a default one that does nothing.

3A: Write a c++ program to add two complex numbers by passing the real and imaginary part of a complex number as parameters to objects and display the summation as result

#include<iostream>

using namespace std;

class complex{

float a, b;

public:

complex(){

a = 0;

b = 0;

}

complex(float x, float y){

a = x;

b = y;

}

void display(){

if(b>=0)

cout<<a<<"+"<<b<<"i"<<endl;

else cout<<a<<b<<"i"<<endl;

}

friend complex sum(complex &A, complex &B){

float a = A.a + B.a;

float b = A.b + B.b;

complex C(a,b);

return C;

}

};

int main()

{

complex A(1.1 , 1.2);

complex B = complex(1.3, 1.4);

complex C;

C = sum(A, B);

C.display();

return 0;

}

**Output:**



3B] Write a C++ program to understand concept of copy constructors(copy content of one object into another)

#include<string.h>

using namespace std;

class copier{

int roll;

string name;

public:

copier(copier& p){

roll = p.roll;

name = p.name;

}

copier(int r, string n){

roll = r;

name = n;

}

void display(){

cout<<"Name "<<name<<endl;

cout<<"Roll No. "<<roll<<endl;

}

};

int main()

{

string name;

int roll;

cout<<"Name: "; getline(cin, name, '\n');

cout<<"Roll No. ";cin>>roll;

copier a(roll,name);

copier b(a);

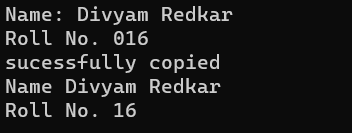
cout<<"sucessfully copied"<<endl;

b.display();

return 0;

}

**Output:**



3C] Write a C++ program to understand concept of dynamic constructor (create the complete class layout)

#include<iostream>

#include<string.h>

using namespace std;

class temp{

char\* word;

int len;

public:

temp(){

word = NULL;

len = 0;

}

temp(char\* w){

len = strlen(w);

word = new char[len+1];

strcpy(word,w);

}

void join(temp p, temp q){

len = p.len + q.len;

word = new char[len + 1];

strcpy(word, p.word);

strcat(word, q.word);

}

void display(){

cout<<word<<endl;

}

};

int main()

{

char\* f1 = "Goa ";

temp name1(f1),

name2("College "), name3("Engineering "), s1, s2;

s1.join(name1,name2);

s2.join(s1,name3);

s2.display();

return 0;

}

**Ouptut:**

****

3D] Write a C++ program to understand concept of destructors in c++ (count of object creation and deletion can be used)

#include<iostream>

using namespace std;

class temp{

int num;

public:

temp(int n){

num = n;

cout<<"Created object "<<num<<endl;

}

~temp(){

cout<<"Destroying object "<<num<<endl;

}

};

int main()

{

cout<<"inside first scope"<<endl;

temp a(1),b(2),c(3);

{

cout<<"inside second scope"<<endl;

temp d(4),e(5);

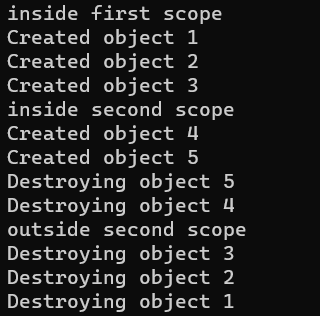
}

cout<<"outside second scope"<<endl;

return 0;

}

**Output;**

****

**Conclusion:** The concepts of Constructors and Destructors in C++ were understood and implemented in the above programs.

**Nitesh Naik**

**(Subject Faculty)**