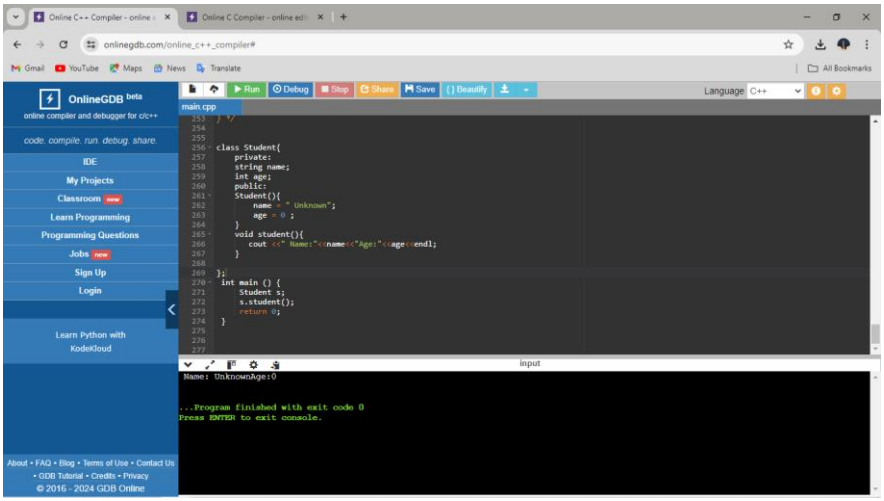
# DAY-2 Assignment (CODES)

## A. Constructors

### 1. Default Constructor:

Write a class Student with a default constructor that initializes the student's name to "Unknown" and age to 0. Add a method display to print the student's details.

```
class Student{
   private:
   string name;
   int age;
   public:
   Student(){
      name = " Unknown";
      age = 0 ;
   }
   void student(){
     cout <<" Name:"<<name<<",Age:"<<age<<endl;
   }
};
 int main () {
    Student s;
    s.student();
    return 0;
 }
```

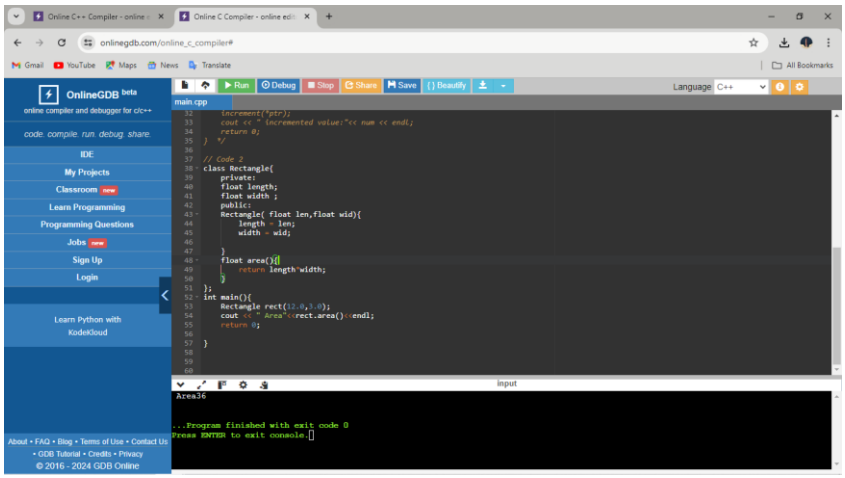



### 2. Parameterized Constructor:

Write a class Rectangle with a parameterized constructor that initializes the length and width. Add a method area that returns the area of the rectangle.

```
class Rectangle{
   private:
   float length;
   float width ;
   public:
```

```
    Rectangle( float len,float wid){
       length = len;
       width = wid;
    }
    float area(){
       return length*width;
    }
};
int main(){
    Rectangle rect(12.0,3.0);
    cout << " Area"<<rect.area()<<endl;
    return 0;
}
```

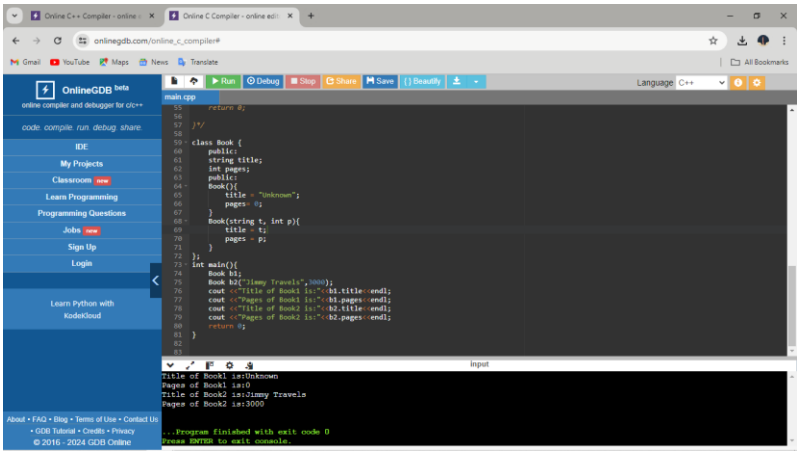



### 3. Multiple Constructors:

Write a class Book that has both a default constructor and a parameterized constructor. The default constructor should set the title to "Unknown" and the number of pages to 0. The parameterized constructor should initialize the title and pages with given values.

```
class Book {
    public:
    string title;
    int pages;
    public:
    Book(){
       title = "Unknown";
       pages= 0;
    }
    Book(string t, int p){
       title = t;
       pages = p;
    }
};
int main(){
    Book b1;
    Book b2("Jimmy Travels",3000);
    cout <<"Title of Book1 is:"<<b1.title<<endl;
```

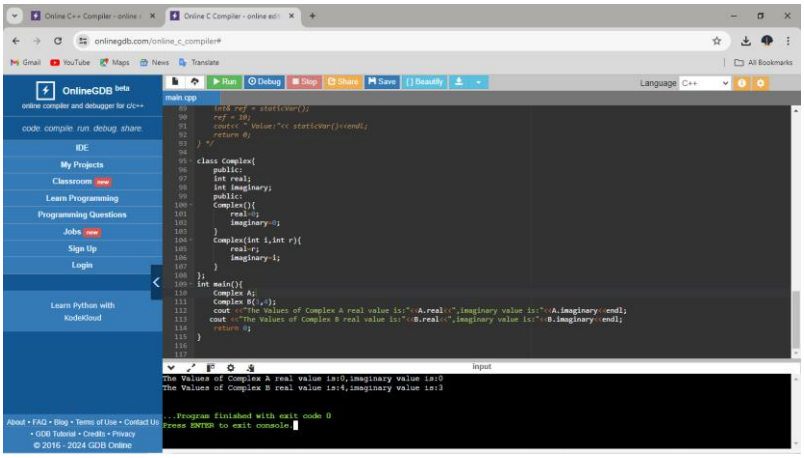
```
    cout <<"Pages of Book1 is:"<<b1.pages<<endl;
    cout <<"Title of Book2 is:"<<b2.title<<endl;
    cout <<"Pages of Book2 is:"<<b2.pages<<endl;
    return 0;
}
```




**4. Constructor Overloading:**

Write a class Complex that represents complex numbers. Implement a default constructor that sets both real and imaginary parts to 0, and a parameterized constructor that takes two arguments to initialize the real and imaginary parts.

```
class Complex{
    public:
    int real;
    int imaginary;
    public:
    Complex(){
        real=0;
        imaginary=0;
    }
    Complex(int i,int r){
        real=r;
        imaginary=i;
    }
};
int main(){
    Complex A;
    Complex B(3,4);
    cout <<"The Values of Complex A real value is:"<<A.real<<",imaginary value is:"<<A.imaginary<<endl;
   cout <<"The Values of Complex B real value is:"<<B.real<<",imaginary value is:"<<B.imaginary<<endl;
    return 0;
}
```

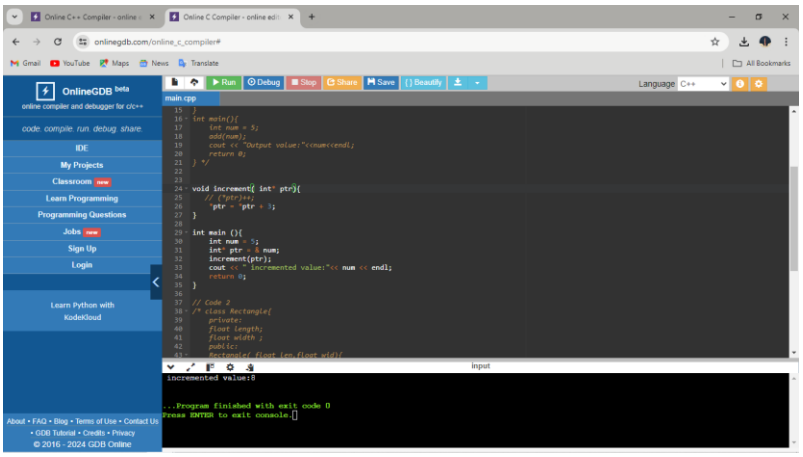



## B. Pointers

### 1. Pointer to an Integer:

Write a function increment that takes a pointer to an integer and increments its value by 1. Demonstrate the function in the main program.

```
void increment( int* ptr){
  // (*ptr)++;
   *ptr = *ptr + 3;
}
int main (){
   int num = 5;
   int* ptr = & num;
   increment(ptr);
   cout << " incremented value:"<< num << endl;
   return 0;
}
```

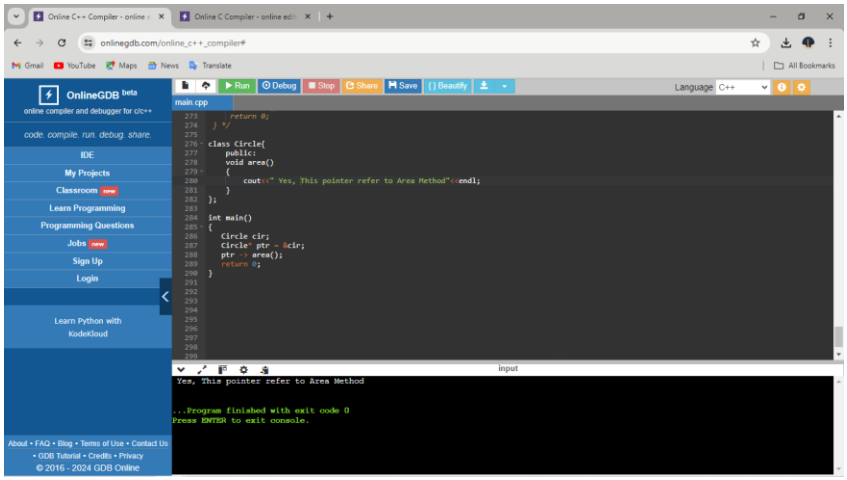



### 2. Pointer to a Class:

Write a class Circle with a method area. Create a pointer to an object of this class and call the area method using the pointer.

```
class Circle{
   public:
   void area()
   {
```

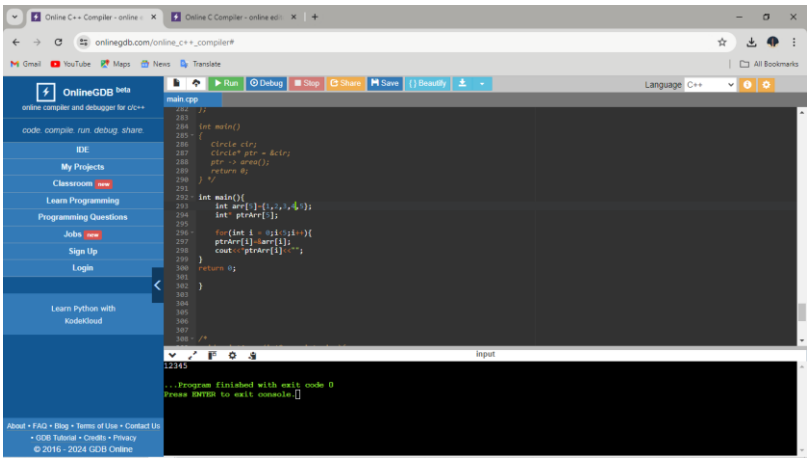
```
        cout<<" Yes, This pointer refer to Area Method"<<endl;
    }
};
int main()
{
  Circle cir;
  Circle* ptr = &cir;
  ptr -> area();
  return 0;
}
```




**3. Array of Pointers:**

Write a program that creates an array of pointers to integers. Initialize the array with values and print them using the pointers.

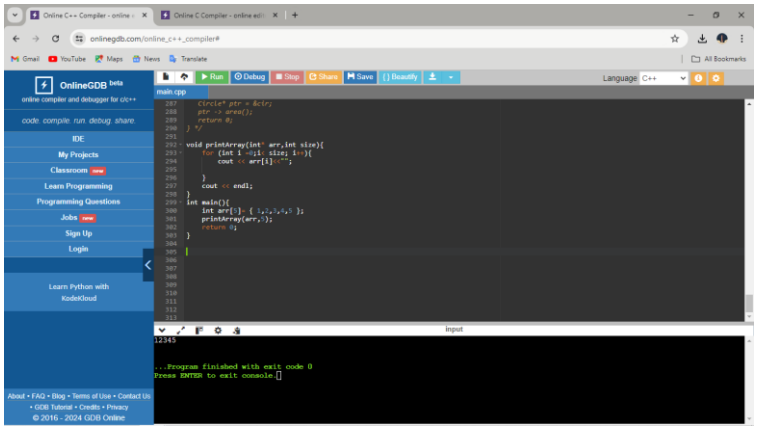
```
int main(){
   int arr[5]={1,2,3,4,5};
   int* ptrArr[5];

   for(int i = 0;i<5;i++){
   ptrArr[i]=&arr[i];
   cout<<*ptrArr[i]<<"";
}
return 0;
```



4. **Pointer to an Array:**

Write a function that takes a pointer to an array of integers and the size of the array. The function should print all elements of the array.

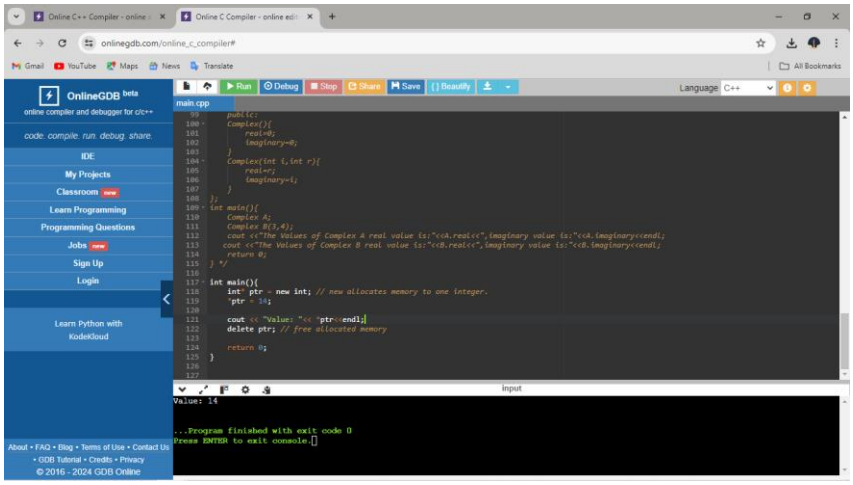
```
void printArray(int* arr,int size){
   for (int i =0;i< size; i++){
      cout << arr[i]<<"";

   }
   cout << endl;
}
int main(){
   int arr[5]= { 1,2,3,4,5 };
   printArray(arr,5);
   return 0;
}
```




5. **Dynamic Memory Allocation: ( Logic )**

Write a program that dynamically allocates memory for an integer, assigns a value to it, and then frees the memory.

```
int main(){
   int* ptr = new int; // new allocates memory to one integer.
   *ptr = 14;

   cout << "Value: "<< *ptr<<endl;
   delete ptr; // free allocated memory

   return 0;
}
```

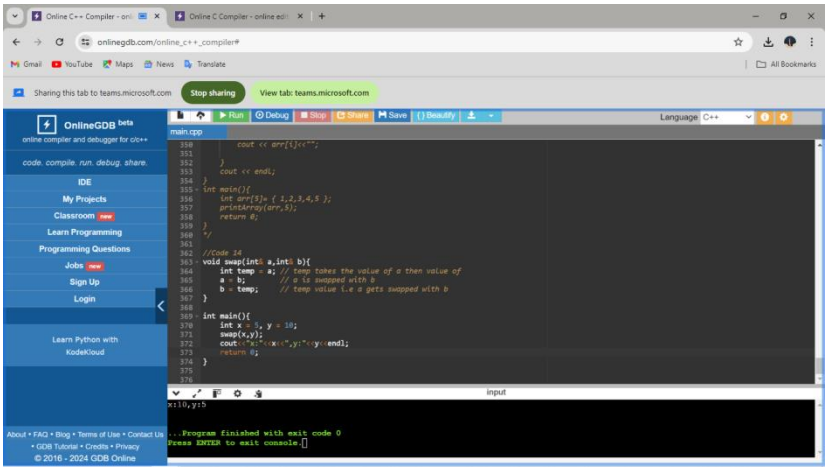



## C. References

### 1. Reference to an Integer:

Write a function swap that takes two integer references and swaps their values. Demonstrate the function in the main program.

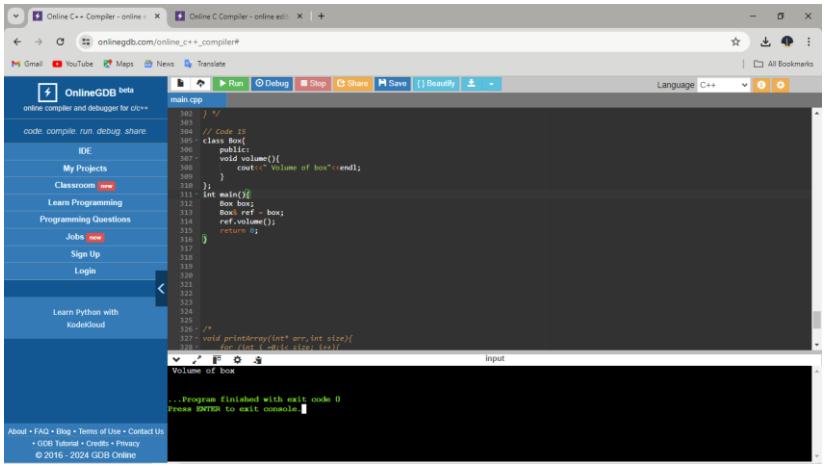
```
void swap(int& a,int& b){
   int temp = a; // temp takes the value of a then value of
   a = b;        // a is swapped with b
   b = temp;     // temp value i.e a gets swapped with b
}

int main(){
   int x = 5, y = 10;
   swap(x,y);
   cout<<"x:"<<x<<",y:"<<y<<endl;
   return 0;
}
```




### 2. Reference to a Class Object:

Write a class Box with a method volume. Create an object of this class and a reference to this object. Call the volume method using the reference.

```
class Box{
   public:
   void volume(){
      cout<< " Volume of Box"<<endl;
   }
};
int main(){
   Box box; // Object of Box class "Box"
   Box& ref = box; // Referencing to class Box
   ref.volume();
   return 0;
}
```




### 3. Returning Reference from a Function: ( Logic )

Write a function that takes an array of integers and returns a reference to the largest element. Demonstrate the function in the main program.

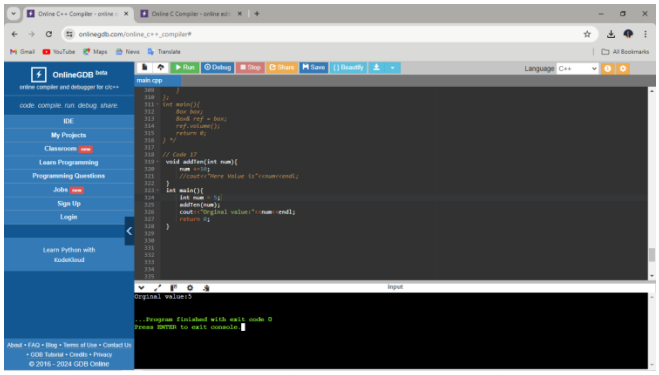
**Logic :** First we need to define a function let's say FunctMaxm and we need to use a loop in it so that max element could be compared and in main we could assign and array let's say Array[] with elements say { 8,2,7,4,6 }then declare a reference to function FunctMaxm . After the function traces out the Max element we could print it using cout.

## D. Pass by Value and Reference

### 1. Pass by Value:

Write a function addTen that takes an integer by value and adds 10 to it. Demonstrate how the original value is not changed after calling the function.

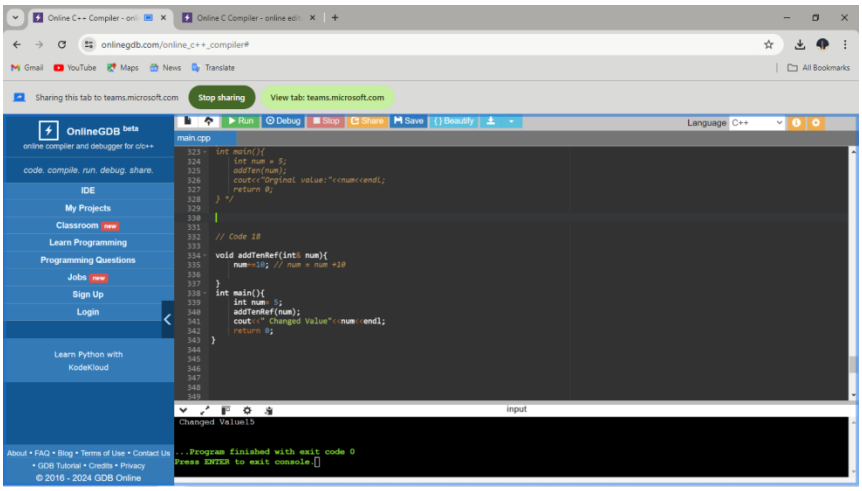
```
void addTen(int num){
   num +=10;
   //cout<<"Here Value is"<<num<<endl;
}
int main(){
   int num = 5;
   addTen(num);
   cout<<"Orginal value:"<<num<<endl;
   return 0;
}
```



**2. Pass by Reference:**

Write a function addTenRef that takes an integer by reference and adds 10 to it. Demonstrate how the original value is changed after calling the function.

```
void addTenRef(int& num){
    num+=10; // num = num +10

 }
 int main(){
    int num= 5;
    addTenRef(num);
    cout<<" Changed Value"<<num<<endl;
    return 0;
}
```

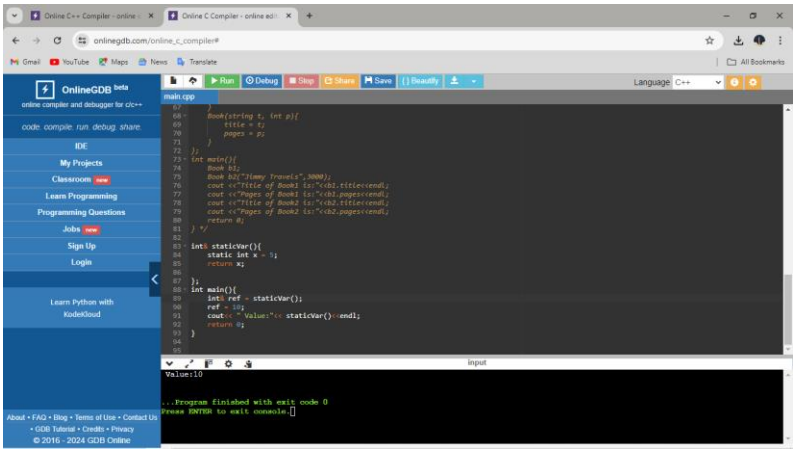


**3. Function Returning a Reference:**

Write a function that returns a reference to a static variable. Modify the returned value in the main function and print it.

```
int& staticVar(){
    static int x = 5;
    return x;

};
int main(){
```

```
    int& ref = staticVar();
    ref = 10;
    cout<< " Value:"<< staticVar()<<endl;
    return 0;
}
```




### 4. Passing Objects by Value and Reference:

Write a class Employee with attributes name and salary. Write two functions: one that takes an Employee object by value and another that takes an Employee object by reference. Modify the salary in both functions and demonstrate the difference in the main program.