**Object Oriented Programming**

**Fall 2022**

|  |  |  |
| --- | --- | --- |
| **Lab 05** | |  |
| **Topic** | **Classes in C++** , **Overloaded Constructors, shallow vs deep copy, copy constructor ,destructor,setterGetters, char \* as attribute, const attributes, Constant objects and constant functions, operator overloading** |
| **Objective** | Making students familiarize with classes and their implementations in  C++, methods, constructors,settersGetters**,** char \* as attributeshallow vs deep copy, copy constructor ,destructor and access Specifiers, const attributes, objects and functions, operator overloading |

**Instructions:**

* Comment your code.
* Use meaningful variable names. .
* Name of the program should be same as the task name. i.e. the first program should be Task\_1.cpp
* **void main() is not allowed. Use int main()**
* **You are not allowed to use system**("**pause**")
* **You are not allowed to use any built-in functions**
* **REMEMBER:** MAKE YOUR OWN DEEP COPY AND STRING LENGTH FUNCTIONS.DONOT USE STRING LIBRARY FUNCTIONS.THEIR DECLARATIONS CAN BE:

**void deepCopy(char\* &dest, const char\* src);**

**int strLength(const char\* src);**

* **Make separate files known as helperFunctions.h and helperFunctions.cpp for strLength and deepCopy functions etc.**
* **You are required to follow the naming conventions as follow:**

**Variables:** firstName; (no underscores allowed)

**Function:** getName(); (no underscores allowed)

**ClassName:** BankAccount (no underscores allowed)

**Students are required to complete the following tasks in lab timings.**

**Task 1:**

We are making a software for a beauty salon. This Salon has a single branch and many male and female employees: for female and for male customers, your software should store Salon Employees Information.

**Salon employees have following members (think for appropriate data types):**

* id (const int) //id should be constant because it can never be changed/updated.
* name (char \*)
* gender (char\*)
* contactNumber (char\*)
* salary (double)
* shift (char\*)

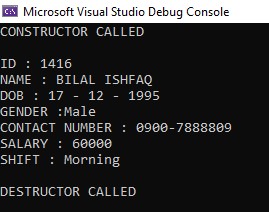
**Now do the following operations on above mentioned class:**

* 1. Write **parameterized constructor with default arguments and copy constructor** (with an output statements to check the life time scope of object)**.**
  2. Write a **destructor** (with an output statement to check the life time scope of object)**.**
  3. Write separate setter (mutator) functions for each attribute to set value (**with no memory leakage**).
  4. Write separate getter (accessor) functions for each attribute to get value (**which should never return the original memory handler**).
  5. Write a **print** function **which should not be the member of the class**. You can define it in source.cpp where you have defined the main() function.

**Use the const qualifier on member functions wherever it is appropriate.**

**Write a tester program that demonstrates capabilities of the class. Make a const object and a non-constant object and call print function for them.**

For example:



**Task 2:**

Define a class **Time** having following private attributes

1. hour (integer)
2. minute (integer)
3. second (integer)

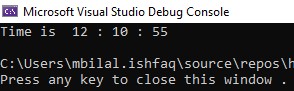
**Now do the following operations on above mentioned class:**

1. Write **parameterized constructor with default arguments** (with input validation)**.**
2. Write separate setter functions for each attribute to set value (with input validation).
3. Write separate getter functions for each attribute to get value.
4. Write a **display** function to display Time with day, month and year on screen.
5. Write operator + to add objects of Time.
6. Write operator – to subtract objects of Time.

**Use the const qualifier on member functions wherever it is appropriate.**

In main() instantiate different objects of Time, add and subtract them and display its information.

**For Example:**



**Task 3:**

You have a class circle having variable radius (in float), default constructor, a constructor having one parameter and behaviors (Area and circumference) with return type as float.

You should implement a copy constructor to perform declaration time initialization.

Driver Program should be like this:

|  |
| --- |
| Circle c1(5);  Circle c2 = c1;  cout<<"Before Setting C2"<<endl;  cout<<"c1.area() = "<<c1.area()<<endl;  cout<<"c1.circumference()= "<<c1.circumference()<<endl;  cout<<"c2.area() = "<<c2.area()<<endl;  cout<<"c2.circumference()= "<<c2.circumference()<<endl;  cout<<"--------------------------------------------"<<endl;  c2.setRad(8);  cout<<"After Setting C2"<<endl;  cout<<"c1.area() = "<<c1.area()<<endl;  cout<<"c1.circumference()= "<<c1.circumference()<<endl;  cout<<"c2.area() = "<<c2.area()<<endl;  cout<<"c2.circumference()= "<<c2.circumference()<<endl; |

**Task 4:**

Run the given below code and make the **void** set\_data(**int**); member function to the constant and try to set the value of x. and then check the behaviour of that , after that make a int x; as constant data member and try to set the value of x using **void** set\_data(**int**);and again check the behavior of the static data members.

|  |
| --- |
| // Example: Constant member function defined outside the class    #include<iostream>  **using** **namespace** std;    **class** Demo  {  **int** x;    **public**:    **void** set\_data(**int**);    **int** get\_data() **const**;    };    **void** Demo::set\_data(**int** a)  {      x=a;  }    **int** Demo::get\_data() **const**   {  **return** x;   }    main()  {      Demo d;      d.set\_data(10);      cout<<endl<<d.get\_data();    **return** 0;  } |

**Output**