OOP

week: 1

21-03-2022

Data Structures

1. CS → Code Segment
2. DS → Data Segment
3. SS → Stack Segment
4. HS → Heap Segment

OOP

Object-oriented programming is centered around the object. Objects are created from abstract data type that encapsulate data and functions together.

Object: An object is a software entity that contains both data and procedures.

Attributes: The data that are contained in an objects are known as attributes.

Member function: The procedures / method that an object performs are called member functions.

Encapsulation: Encapsulation refers to the combining of data and code into a single object.

Data hiding: Data hiding refers to an objects ability to hid its data from code that is outside the object. Only the object's member functions may directly access and make changes to the objects data.

Instance Each object that is created from a class is called an instance.

Access Specifiers: The keyword 'Public' and 'Private'

are known as access specifiers because they specify how class members may be accessed

Getters: The functions which returns the value of 'private' attributes. In has same ~~void~~ data type as the attribute has.

Setters: The functions which sets the value of Private attributes. In has void data type.

- The prototype of member functions are written in class declaration.
- But the defination can be and cannot be written inside class declaration.
- The defination written outside the class declaration use a operator (::) called scope resolution operator. It identifies the function as a member of a Rectangle class

ReturnType classname :: functionName() any name.

Accessors: A member function that gets a value from a class's member variable but does not change it is known as an accessor. e.g getters.

Mutators: A member function that stores a value from a class's member variable & and also change its value is known as mutators. e.g. setter

Instantiation: Defining a class object is called.

```
Class Rectangle { };
int main() {
    Rectangle rec1;   ← Instantiation
}
```

Rectangle → class, Instance; rec1 → object

Date: 5-4-2022

• Memory Managment

- C++ Memory managment
- Explicit

• Unique Pointer

#include <memory> — header file for unique ptr

Syntax:

unique_ptr < Rectangle > rectangle Ptr (new Rectengle);
(Rectangle: class; rectangle Ptr: name; Rectengle: class object)

• exit ( )

- exit the program compilation
- EXIT_FAILURE = return -1 — program completed unsuccessfully

Rectangele * rectPtr = nullptr;
(Rectangele: class; rectPtr: pointer; nullptr: ↓ This is a keyword which assign null value to pointer.)

↓ This pointer can also hold the address of Rectangle ~~Objects~~ Object.

"->" —> This operator is also used to derefrence the pointer.

Inline member function

When a member function is defined in the declaration of a class, it is called an inline function. When function is inline function So there is no need to use scope resolution operator.

Constructor

A constructor is a member function that is automatically called when a class object is created

- It is automatically called.
- It has no datatype and it has the same return type as the class.
- It is helpful to used as initilizing the attributes of class.
- Constructors are not executed by explicit function calls and they do not return a value.

There are two types of constructor

(1) Default Constructor

- The Constructor that do not take arguments.
- When any object is created by dynamic memory allocation (new operator) its default constructor is automatically executed.

(2) Parametrized Constructor

- The Constructor that takes any argument is called Parametrized Constructor.
- It is called by using / passing an argument.
- It also works like common functions that take arguments.

Destructors

A destructor is a member function that is automatically called when an object is destroyed.

- Same name as class
- Preceded by a tilde character (~) e.g. ~Rectang
- Perform shut down procedures

- Destructors have no return type.
- Destructors cannot accept arguments, so they never have parameter list.

Overloading Constructor

- When two or more functions have same name they are overloaded same is the case with Constructors.
- But there Parameter lists are different
- Number of Parameters or Pat Parameter data type
- A program only have one default constructor.
- Parametrized Constructor with default argument also work as default constructor if no arguments are given.

Array of Objects

- Arrays of objects also exists
- They work like common arrays.
- Default constructor is called for each object in the array.

Example:

Inventory[] = {"Hammer", "Wrench", "Pliers"}

Index	
0	Hammer
1	Wrench
2	Pliers

- If there are fewer initializers in the list than object in the arrays, the default constructor will be called for all the remaining objects.

Instance

- Each class object is an instance of the class
- Each instance of a class has its own copies of the class instance variables.

Static Variable and functions

- If a member is declared static, all instance of that class have access to that variable. If a member function is declared static, it may be called without any instance of the class being defined.

MIDS

Date : 16-5-2022

Friends of Classes

- A friend is a function or class that is not a member of a class, but has access to the private member of the class.
- Syntax

  friend ReturnType FunctionName (Parameter List)

- It can be any function, or it can be a member of another class
- A class can be declared a friend of another class.

Memberwise Assignment

- The = operator may be used to assign one object's data to another ~~data~~ object, or to initialize one object with another object's data. By default, each member of one object is copied to its counterpart in the other object.

=> Operator Overloading

- Use of standerd operator with class objects.

Date: 23-5-22

OOP design

- **Relationships**

① Composition (has a) | (whole part)

- class A is part of class B

↓

Composition of

- whole is composed of other objects
- whole is itself nothing • part cannot survive outsid

② Aggregation

- Almost same as composition
- Part can survive outside the whole

③ Association

- Weekest link between two objects

Memory management in C++

When a program is loaded into memory, it is organized into three areas of memory called segments. They are given below:

① Text segment (Code segment) — where the compiled code of program resides.

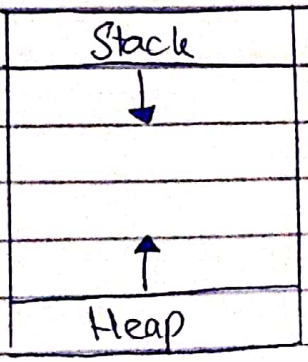
② Heap segment — when program allocate memory at runtime

- Heap segment grows upward when more memory is needed or allocated.

③ Stack segment — where the local variables are stored and is used for passing arguments to the functions along with return address of the instruction

④ There is another segment called code segment. It in divided into two parts

- Initialized data segment — All global, static and constant data are stored in data segment.
- Uninitialized data segment — All the uninitialized data are stored in BSS.



- Stack is grown downward
- Heap is grown upword