C++ Programming Review

CS130A

by
Omid Askari
Victor Zakhary

Introduction to C++

- Functional Programming
- Pointers and References
- Arrays
- Object Oriented Programming
- Template
- Call by Value, Pointer and Reference
- Previous data structures

Functional (Procedural) Programming

C++ language supports both functional and object oriented programming paradigm (muli-paradigm).

In functional programming, we can define different functions and use them in each other. The only constraint is

You cannot use a function before its implementation. In fact, there is an order in C/C++ for using functions. We also can have recursive functions in C++.

Git

Pointers and References

Reference is just an address of a memory point (variable). It can be obtained by ampersand operator (&). type& name;

Pointer is a variable that stores memory address or in fact the address of another variable. It should have type and is defined as: type* name;

Pointers are more flexible and have less limitations.

Differences between pointers and references: Git

1. We use pointer syntax to access the values "pointed to" by the pointer.
2. We can redirect the pointer to point it to a different "target" variable.
3. We can make a pointer point to nothing (ie, NULL pointer).

Common mistakes with pointers

- Different types of dangling pointers
- Memory Leaks
- Incorrect deletion

Git

Arrays

- Static arrays:

  Their size is constant and should be known at compile time.

  ```
  double a[100];
  ```

- Dynamic arrays:

  Their size can be defined at running time.

  ```
  double *a = new double[leng];
  ```

  We can also use pointers to iterate within dynamic arrays. Git

Call by Value, Pointer and Reference

by value:

```
void function( T t1, T t2 )
```

by reference:

```
void function( T& t1, T& t2 )
```

by pointer:

```
void function( T* t1, T* t2 )
```

Git

Object Oriented Programming

- Class
- Struct
- Object
- Access modifiers (private, public and protected)
- Constructor
- Destructor
- What are default, parametric, conversion and copy constructors?

Git

Template

Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type.

It is defined as

```
template <typename T>        or        template <class T>
```

for instance:

```
T const& max(T const& a, T const& b){
        return a < b ? b : a;
}
```

Git

Previous data structures

Stack: Last In First Out (LIFO) policy

Queue: First In First Out (FIFO) policy

They can be implemented by vector, linked lists and array.

Git

Object Oriented Programming: Advanced Concepts

- Encapsulation
- Inheritance
- Polymorphism
- Static function and static variable
- Virtual function and const function
- Overloading operators

Any Questions ?