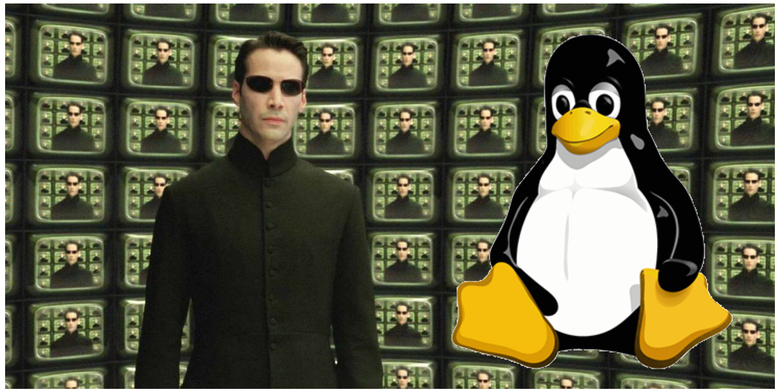
16.07.20 15:58

CSCI 125
Introduction to Computer Science and Programming II
Lecture 6: User Class II



Jetic Gū

2020 Summer Semester (S2)

Overview

- Focus: Basic C/C++ Syntax
- Architecture: Linux/Unix OS
- Core Ideas:
    1. Pointer Operations of Class: new and delete, destructor
    2. Tutorial: C++ list class

P0
Review

Data Types

- Primary √
  - Integers, Characters, Boolean √
  - Floating point √
  - Void√
- Derived
  - Function√, Array√, Pointer, Reference√
- User Defined
  - **Struct, Class**, Enumerate√, Typedef√

Review

Class Pointer Operations

There are a few different things

Summary

P1
Class Pointer

C++ Class

- User defined data types
- Members: variables and function
  - access specifiers
- Constructors

Review

P2
Constructor

Destructors of Class

- Stuff to do when a class instance is "destructed"
  - When exiting the scope
  - The variable is manually deleted
- Download H805
  - Look at `demo1.cpp`

```
3.  class Example {
4.   public:
5.       Example();
6.       ~Example();
7.  };
8.  Example::Example() {
9.       cout << "Constructed!" << endl;
10. }
11. Example::~Example() {
12.      cout << "Destroyed!" << endl;
13. }
```

Technical

Pointer Class Instances

- Declaration
  - ClassName * variableName;
- What is this?
  - A pointer variable: a small memory space is reserved for a memory address
  - No further memory for the **actual instance** has been allocated yet!

Concept

P1
Class Pointer

Pointer Class Instances

```
1. string *str;
```

- This variable store a memory address, that is going to be treated as the address of a `string` instance object
- How do we manually allocate space for it?

Concept

P1
Class Pointer

Pointer Class Instances

```
ClassName * p;

p = new ClassName;
```

- Allocate memory space for a ClassName instance, and return it's address to `p`

```
delete p;
```

- Recycle the memory space starting at address `p`

Concept

H805 Demo2

- Download H805
- Look at `demo2.cpp`
- Line5: `new string`
  - This will **call the constructor**!
- Line10: `delete str`
  - This will **call the destructor**!

```
5. string *str = new string;
6. cin >> *str;
7. cout << "add: " << str;
8. cout << "; value: " <<
   *str;
9. cout << endl;
10.delete str;
```

Accessing Members

```
string * p = new string;
p.c_str(); // this will NOT work
```

- `p` is a pointer now, it doesn't have member function `c_str`
- How do we access members of a pointer class variable?

```
p->c_str(); // this will work
```

Concept

H805 Demo3

- Download H806
- Look at `demo3.cpp`
- Line7: does not work
- Line8:
  - `str` is a pointer
  - `c_str` is a member function of string objects
  - access: `Pointer->Member`

```
5. string *str = new string;
6. cin >> *str;
7. // cout << str.c_str();
8. cout << str->c_str();
9. cout << endl;
10.delete str;
```

Accessing Members

expression	can be read as
`*x`	pointed to by `x`
`&x`	address of `x`
`x.y`	member `y` of object `x`
`x->y`	member `y` of object pointed to by `x`
`(*x).y`	member `y` of object pointed to by `x` (equivalent to the previous one)
`x[0]`	first object pointed to by `x`
`x[1]`	second object pointed to by `x`
`x[n]`	`(n+1)` th object pointed to by `x`

Concept

1. http://www.cplusplus.com/doc/tutorial/classes/

`this` pointer

- Each object also has a `this` pointer, which is `private`
  - Gives members information on this object's address

```
3.  class MyClass {
4.   public:
5.      MyClass* add()
6.          {return this;}
7.  };


10. MyClass x;
11. cout << (&x == x.add()) << endl;
```

Concept

Tutorial: C++ list class

Objective

- In Python list, you can perform 2 operations that are quite handy
  - `someList.insert(0, x)`
    additional element at `someList[0] == x`, and move all subsequent stuff rightwards
  - `someList.append(x)`
    additional element at `someList[-1] == x`
  - You can always add stuff to `someList` until you run out of memory

Review

P2
Tutorial: List

C++

- C++ arrays
  - Cannot append elements
    unless there’s enough space
  - Cannot prepend elements
    You have to move everything around to make this happen, this takes time
  - Fixed length from declaration

Review

Objective Class: MyList

- Use pointers to chain objects
- Support prepend and append operations
- Support indexed access and access using index -1 (last element)

Concept

P2
Tutorial: List

Objective Class: MyList

- Chained elements
  `next` points to the next element in the list;
  `last` points to the last element in the list;

- 10: Get value at index `ind`

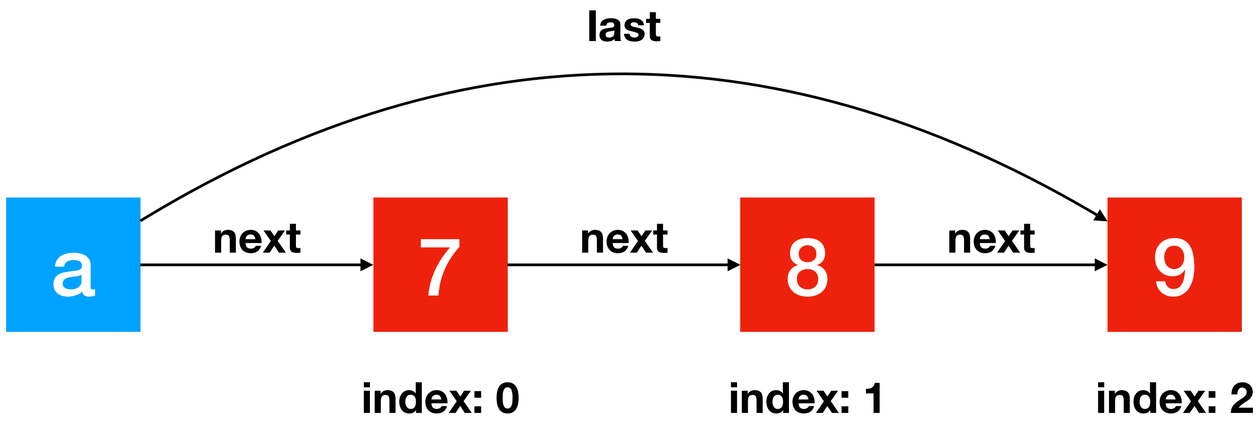
- 11: Write value `val` to index `ind`

```
1. class MyList {
2.      int value;
3.      MyList* next;
4.      MyList* last;
5. public:
6.      int length;


8.      void prepend(int val);
9.      void append(int val);
10.     int get(int ind);
11.     int give(int ind, int val);
12.     MyList(); ~MyList();
13.};
```

Technical

Objective Class: MyList



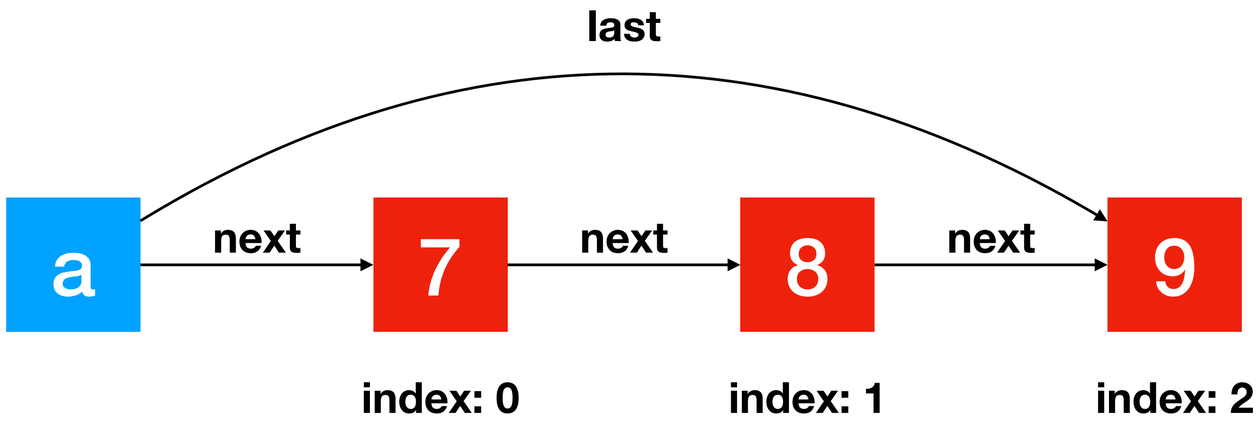

```
1. MyList a;
2. a.append(7);
3. a.append(8);
4. a.append(9);
```

Object

Pointer Object

Technical

Objective Class: MyList




```
5. a.get(2); //
```
access index 2

1. go to `a.next`;
2. go to `a.next->next`;
3. go to `a.next->next->next`;