UCLA Samueli Computer Science

CS32: Introduction to Computer Science

Discussion Week 2

Junheng Hao, Ramya Satish

Jan 18, 2019

Hello, CS32!

UCLA Samueli Computer Science

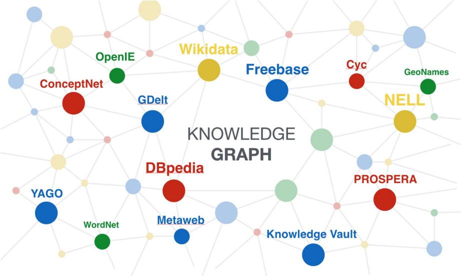
- Course website: http://web.cs.ucla.edu/classes/winter19/cs32/
- CS Discussion 2B
  - Time and place: 12:00-2:00pm, Friday @Rolfe Hall 3134
  - Discussion sessions on week 2, 4-10
- TA: Junheng Hao
  - Email: haojh.ucla@gmail.com
  - Office Hours: Thursday 8:30am-9:30am & 11:30am-1:30pm @BH3256S
- LA: Ramya Satish
  - Email: ramyasatish1997@ucla.edu
  - Office Hours: Thursday 8:30am-9:30am & 4:30pm-5:30pm @BH3256S
- Discussion Website: https://www.haojunheng.com/teaching/cs32-winter19/

  (Slides will be posted here!)

About TA (Junheng Hao)

- I am a second-year PhD student at ScAi Data Mining Lab, Department of Computer Science, working with Yizhou Sun and Wei Wang.
- Other courses I taught: CS145 (Intro to data mining)
- Research Interests:
    - Knowledge Graph & Knowledge Bases
    - Natural Language Processing
    - Graph Mining
    - Healthcare + AI
- I am also learning languages (German & Spanish) for research and for fun!

Taken on August 14, 2017 at
Wadi rum, Jordan
Credit: Bo Chen (UBC)



More about my research

- Mining knowledge graphs → AI + KG + NLP (+ Healthcare)
    - Multi-view knowledge graphs (inferences on Wiki ontology and instances)
    - Multilingual knowledge graphs (cross-lingual search)
    - Healthcare: Automatic analyzer reports via medical knowledge bases

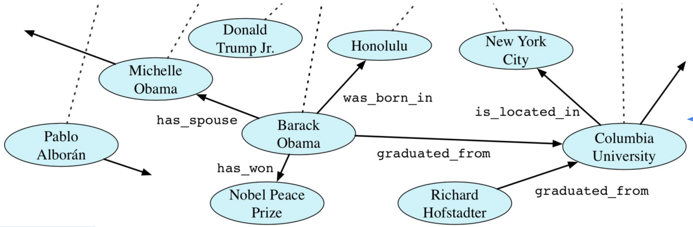
- If you would like to join us in AI & Data Mining group as an undergrad researcher, contact me or Prof. Yizhou Sun.
    - Definitely, you are also welcomed to our lab in later quarter if you need more time to learn basic machine learning knowledge. :)

More about my research

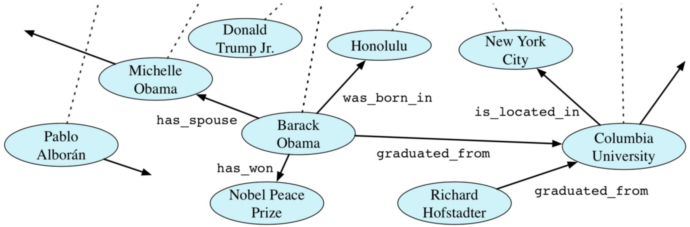
UCLA Samueli Computer Science

→ Knowledge Graphs

English Knowledge Graph



German Knowledge Graph



→ Description

Alain Connes

French mathematician, currently Professor at the Collège de France, IHÉS, The Ohio State University and Vanderbilt University. He was an Invited Professor at the Conservatoire national des arts et métiers (2000)

Alain Connes

Alain Connes, ist ein französischer Mathematiker und Träger der Fields-Medaille.

→ Word

French ● ● ●

Mathematician ● ● ●

französischer ● ● ●

Mathematiker ● ● ●

About LA (Ramya Satish)

UCLA Samueli Computer Science

- Undergraduate senior in Computer Science
- Took CS 32 in Winter 2016
  - My favorite CS class at UCLA :)
- Pursuing software engineering
- UCLA activities
  - UPE
  - MentorSEAS
  - TBP
  - SWE
- Other interests
  - Learning languages: French & Tamil, just started learning Korean
  - Netflix, music, napping

About the course

What you will learn in this course…

More about C++ and data structures. Well, that’s a lot. And very important.

- Review of C++
- **Object-oriented programming:** Data abstraction, Inheritance and Polymorphism, Recursion, …
- **Data structures:** Arrays, Lists, Trees, Graphs, Hash tables, …
- **Algorithms** (eg. sorting) **and complexity analysis**

Announcement

- Check course website: http://web.cs.ucla.edu/classes/winter19/cs32/
- Homework & Project:
    - Homework 1 due on Tuesday, January 22
    - Project 2 due on Tuesday, January 29
- There is no discussion in Week 3 (Jan. 25). Prof. Smallberg will have lectures instead.
- Midterm 1: Jan 30

About the discussion

- Part 1: Learning (Junheng) ~50 minutes
    - Review some topics, questions and examples based on last week’s lectures
    - Some extended topics from lectures

```
// A break of 5-10 minutes (This is not commented)
```

- Part 2: Exercise (Ramya) ~50 minutes
    - Hands-on Group exercise questions

Outline Today

- Dynamic memory allocation in C++
- Constructor, destructor, copy constructors, and assignment operators
- Data structure: Arrays & Linked Lists (probably just start)

Dynamic memory allocation

Compare: Static Memory Allocation

If we want to type in a paragraph and save it into a C-string.

```
#define MAXLENGTH 10000
char s[MAXLENGTH+1];
cin.getline(s);
```

- What if the paragraph is extremely long? → Out-of-bound
- What if the paragraph has only five words? → Overallocated Memory (Waste)

Dynamic memory allocation

`new` in C++

What if we want to fit the paragraph into a C-string with right the sufficient size?

```
<type> *<name> = new <type>[<#elements>];
char *article = new char[length];
```

Unsigned int variable

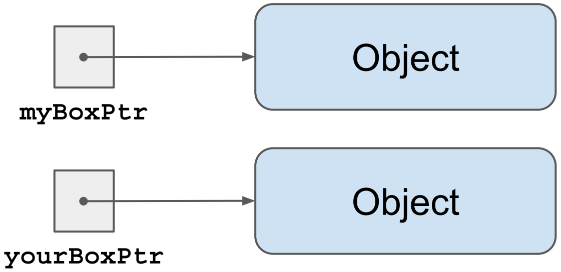
`new` will dynamically allocate the sequential memory space for the requested data type and size and return the starting address of the allocated memory space.

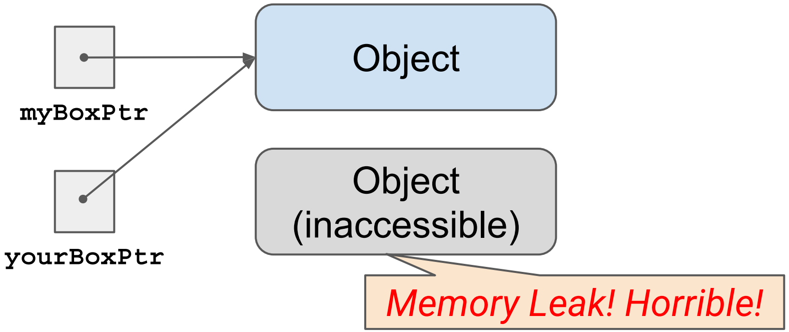
Variables allocated with **`new`** will remain in the memory unless we manually delete it which means dynamic allocation has to be deleted once we no longer use it.

```
delete [] article
```

Dynamic memory allocation

Memory Leak



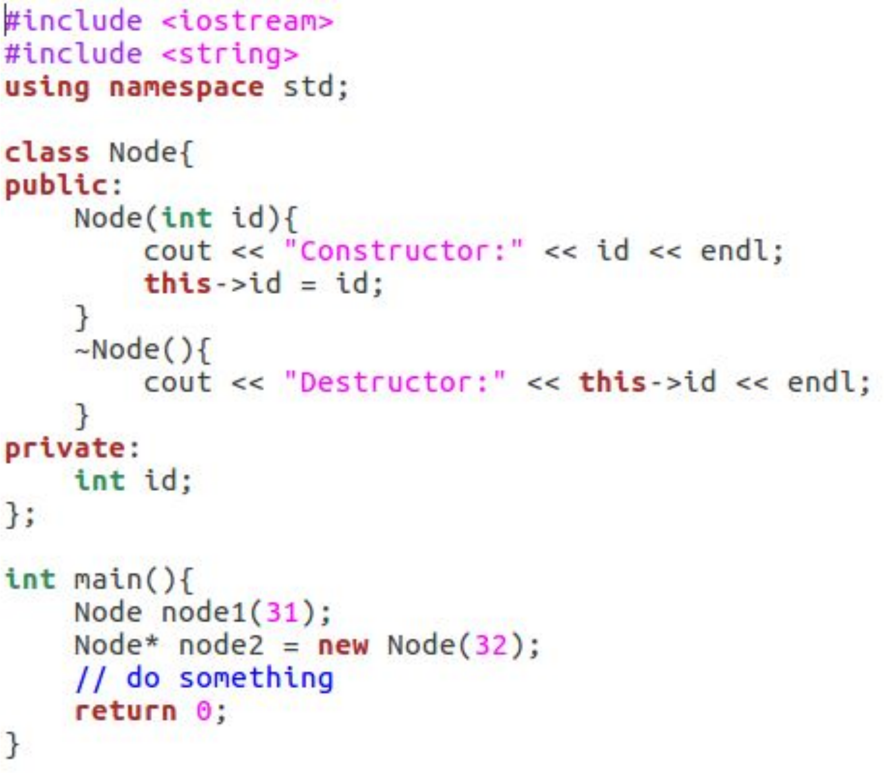


Memory Leak! Horrible!

```
// Create first object
MagficBox* myBoxPtr = new MagficBox

// Create second object
MagficBox* yourBoxPtr = new MagficBox

// Assignment causes an inaccessible object
yourBoxPtr =  myBoxPtr
```

Dynamic memory allocation

A Problematic Code (1)

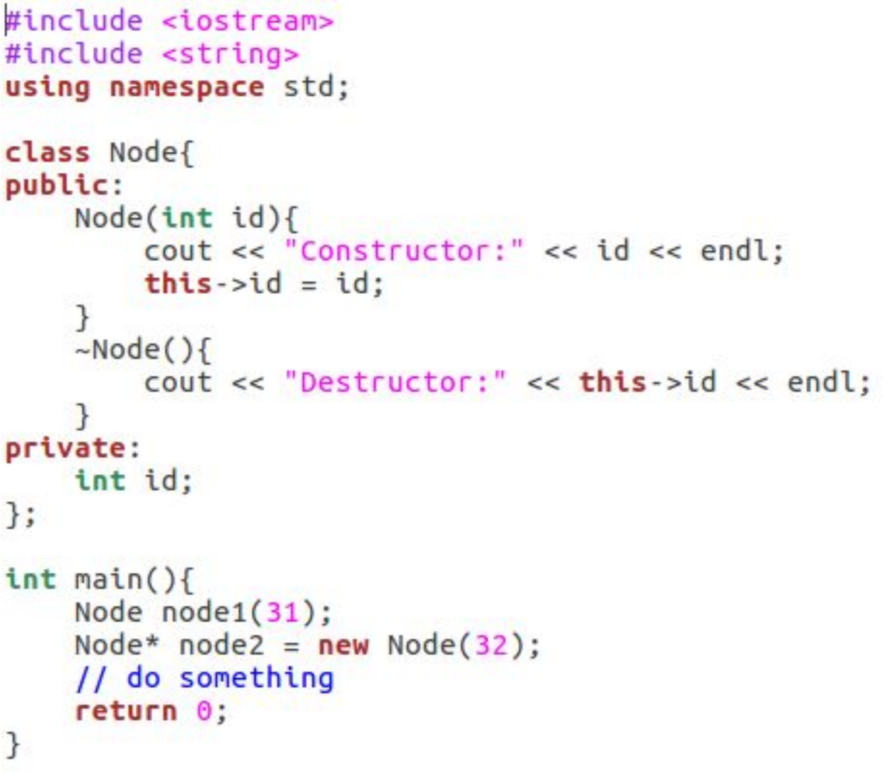


```cpp
#include <iostream>
#include <string>
using namespace std;

class Node{
public:
    Node(int id){
        cout << "Constructor:" << id << endl;
        this->id = id;
    }
    ~Node(){
        cout << "Destructor:" << this->id << endl;
    }
private:
    int id;
};

int main(){
    Node node1(31);
    Node* node2 = new Node(32);
    // do something
    return 0;
}
```

What is the output of this simple “node” program?

Dynamic memory allocation

A Problematic Code (1)



```
#include <iostream>
#include <string>
using namespace std;

class Node{
public:
    Node(int id){
        cout << "Constructor:" << id << endl;
        this->id = id;
    }
    ~Node(){
        cout << "Destructor:" << this->id << endl;
    }
private:
    int id;
};

int main(){
    Node node1(31);
    Node* node2 = new Node(32);
    // do something
    return 0;
}
```

What is the output of this simple “node” program?

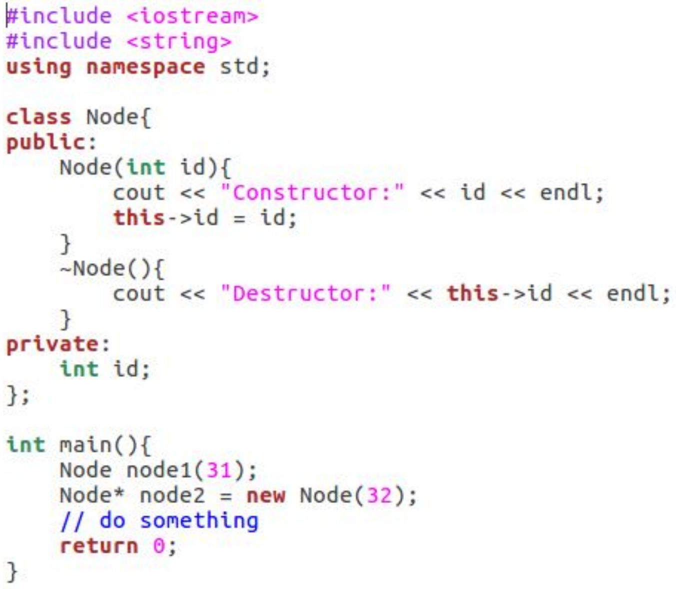
```
Constructor:31
Constructor:32
Destructor:31
```

→ Node(32) does not deconstruct!

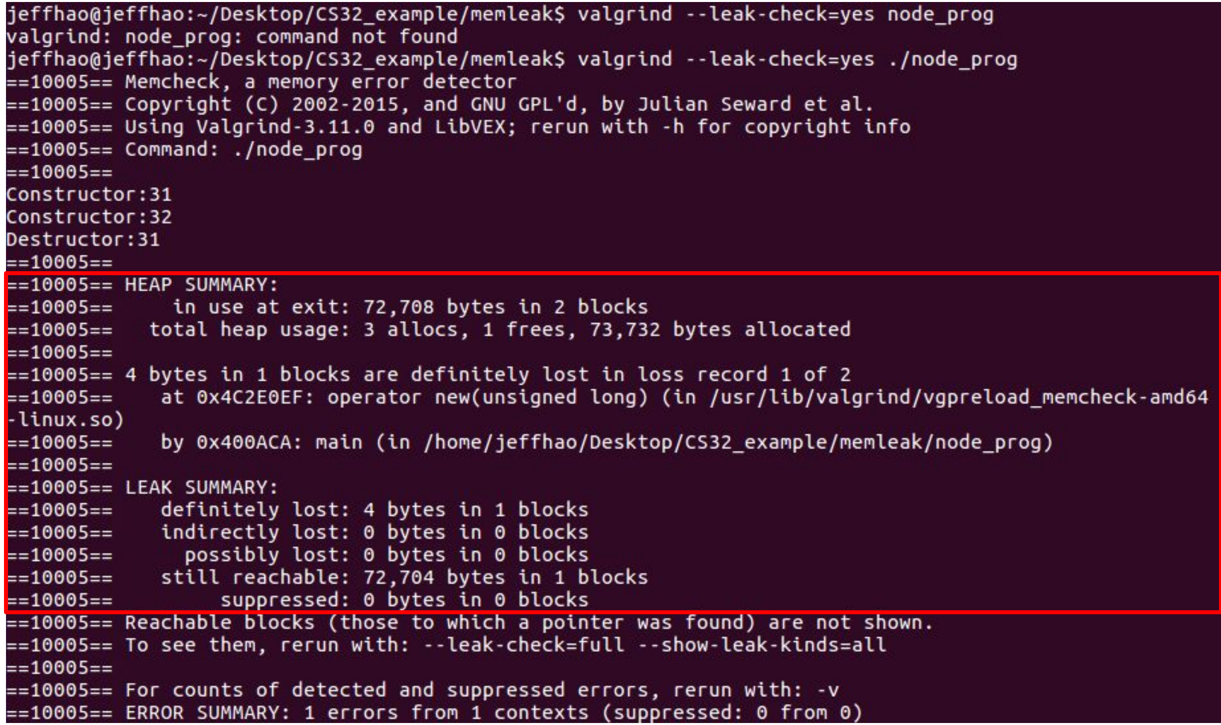
Dynamic memory allocation

A Problematic Code (2)

Let’s check it by Valgrind!



```cpp
#include <iostream>
#include <string>
using namespace std;

class Node{
public:
    Node(int id){
        cout << "Constructor:" << id << endl;
        this->id = id;
    }
    ~Node(){
        cout << "Destructor:" << this->id << endl;
    }
private:
    int id;
};

int main(){
    Node node1(31);
    Node* node2 = new Node(32);
    // do something
    return 0;
}
```



```
jeffhao@jeffhao:~/Desktop/CS32_example/memleak$ valgrind --leak-check=yes node_prog
valgrind: node_prog: command not found
jeffhao@jeffhao:~/Desktop/CS32_example/memleak$ valgrind --leak-check=yes ./node_prog
==10005== Memcheck, a memory error detector
==10005== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==10005== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==10005== Command: ./node_prog
==10005==
Constructor:31
Constructor:32
Destructor:31
==10005==
==10005== HEAP SUMMARY:
==10005==     in use at exit: 72,708 bytes in 2 blocks
==10005==   total heap usage: 3 allocs, 1 frees, 73,732 bytes allocated
==10005==
==10005== 4 bytes in 1 blocks are definitely lost in loss record 1 of 2
==10005==    at 0x4C2E0EF: operator new(unsigned long) (in /usr/lib/valgrind/vgpreload_memcheck-amd64
-linux.so)
==10005==    by 0x400ACA: main (in /home/jeffhao/Desktop/CS32_example/memleak/node_prog)
==10005==
==10005== LEAK SUMMARY:
==10005==    definitely lost: 4 bytes in 1 blocks
==10005==    indirectly lost: 0 bytes in 0 blocks
==10005==      possibly lost: 0 bytes in 0 blocks
==10005==    still reachable: 72,704 bytes in 1 blocks
==10005==         suppressed: 0 bytes in 0 blocks
==10005== Reachable blocks (those to which a pointer was found) are not shown.
==10005== To see them, rerun with: --leak-check=full --show-leak-kinds=all
==10005==
==10005== For counts of detected and suppressed errors, rerun with: -v
==10005== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

Dynamic memory allocation

Good memory example from textbook

```
/** @file GoodMemory.cpp */
# include “GoodMemory.h”

GoodMemory::GoodMemory() : someBoxPtr(nullPtr)
{
} // end default constructor

GoodMemory::~GoodMemory()
{
    delete someBoxPtr;
} // end constructor

void GoodMemory::unleakyMethod(const double& someItem)
{
    someBoxPtr = new ToyBox<double>();
    someBoxPtr->setItem(someItem)
} // end unleakyMethod
```

Constructor & Destructor

Class Composition

Class composition is when a class contains one or more member variables that are objects.

Order of construction:

- Member variables are constructed in order.
- Then the current class constructor is executed.

Order of destruction:

- The current class destructor is executed first.
- The member variables are destructed in the reverse order.

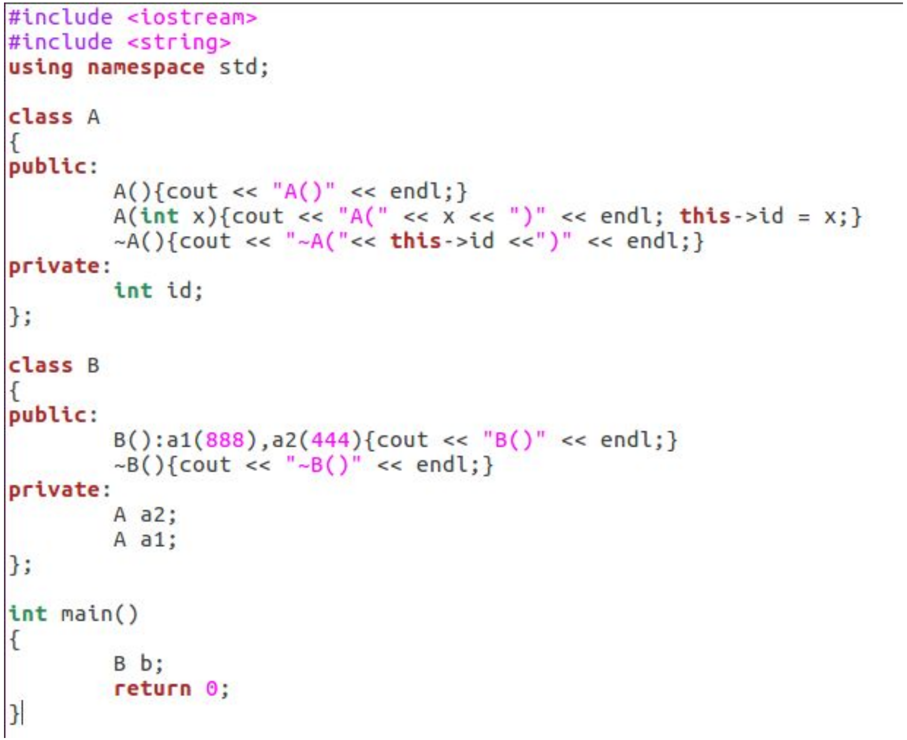
Summary:

- Constructor: Inside -> Outside (In-order)
- Destructor: Outside -> Inside (Reverse Order)

Constructor & Destructor

Question: For classes containing members of a class type, what is the order of construction and destruction?

What is the output?



```
#include <iostream>
#include <string>
using namespace std;

class A
{
public:
        A(){cout << "A()" << endl;}
        A(int x){cout << "A(" << x << ")" << endl; this->id = x;}
        ~A(){cout << "~A("<< this->id <<")" << endl;}
private:
        int id;
};

class B
{
public:
        B():a1(888),a2(444){cout << "B()" << endl;}
        ~B(){cout << "~B()" << endl;}
private:
        A a2;
        A a1;
};

int main()
{
        B b;
        return 0;
}
```

Constructor & Destructor

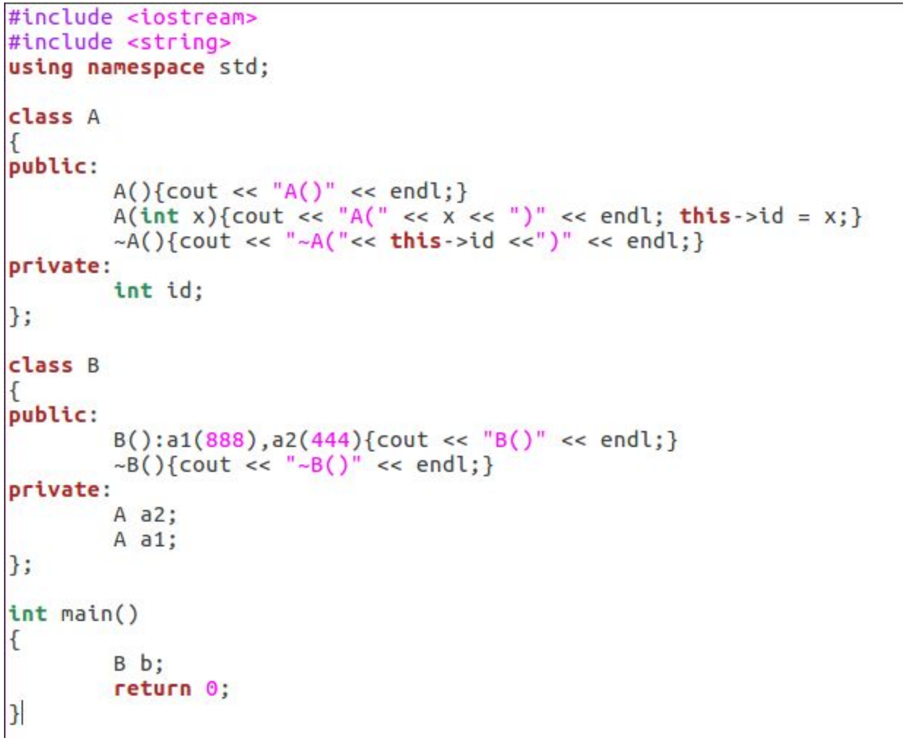
UCLA Samueli Computer Science

Question: For classes containing members of a class type, what is the order of construction and destruction?

What is the output?

Answer:

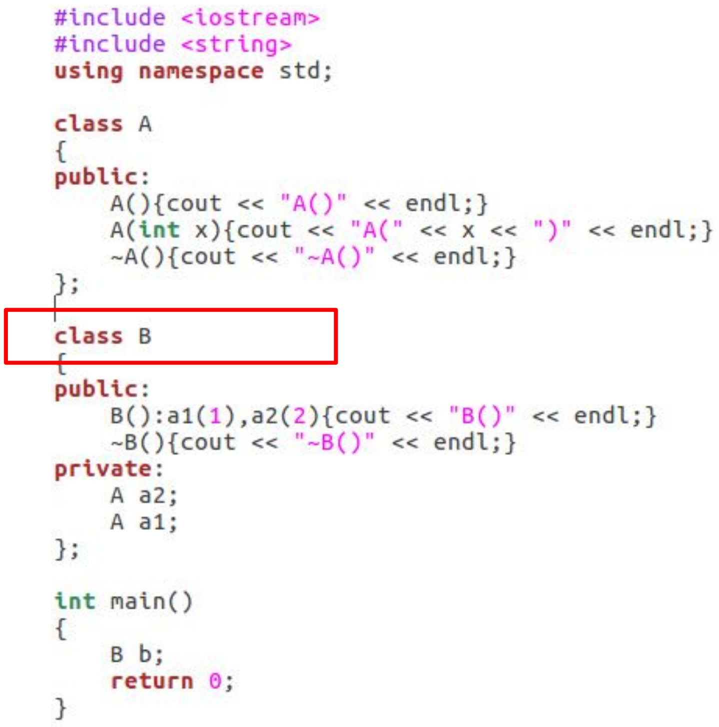
```
A(444)
A(888)
B()
~B()
~A(888)
~A(444)
```



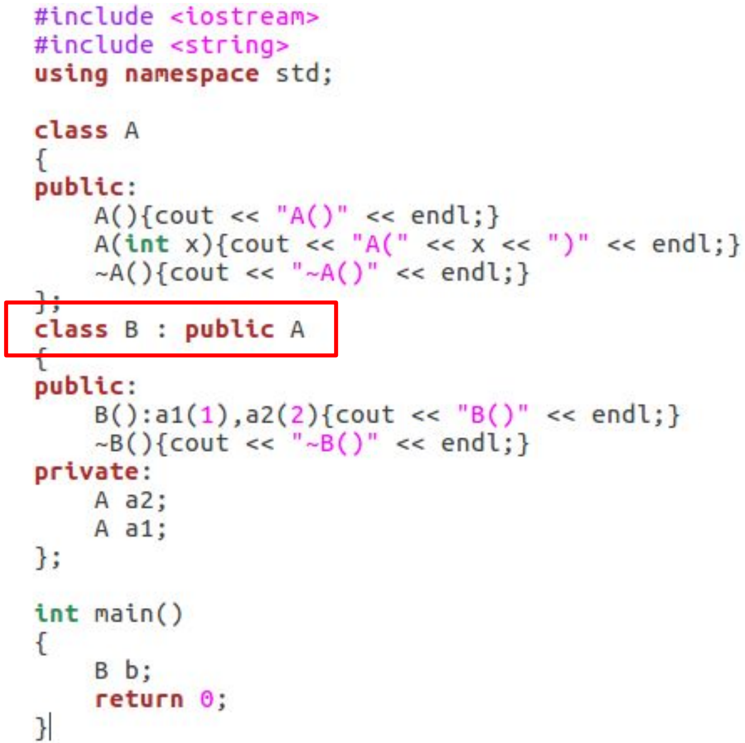
```cpp
#include <iostream>
#include <string>
using namespace std;

class A
{
public:
        A(){cout << "A()" << endl;}
        A(int x){cout << "A(" << x << ")" << endl; this->id = x;}
        ~A(){cout << "~A("<< this->id <<")" << endl;}
private:
        int id;
};

class B
{
public:
        B():a1(888),a2(444){cout << "B()" << endl;}
        ~B(){cout << "~B()" << endl;}
private:
        A a2;
        A a1;
};

int main()
{
        B b;
        return 0;
}
```

Constructor & Destructor

Note: There is a difference between class composition and class inheritance. → Will be explained in later lectures.



```
#include <iostream>
#include <string>
using namespace std;

class A
{
public:
    A(){cout << "A()" << endl;}
    A(int x){cout << "A(" << x << ")" << endl;}
    ~A(){cout << "~A()" << endl;}
};

class B
{
public:
    B():a1(1),a2(2){cout << "B()" << endl;}
    ~B(){cout << "~B()" << endl;}
private:
    A a2;
    A a1;
};

int main()
{
    B b;
    return 0;
}
```



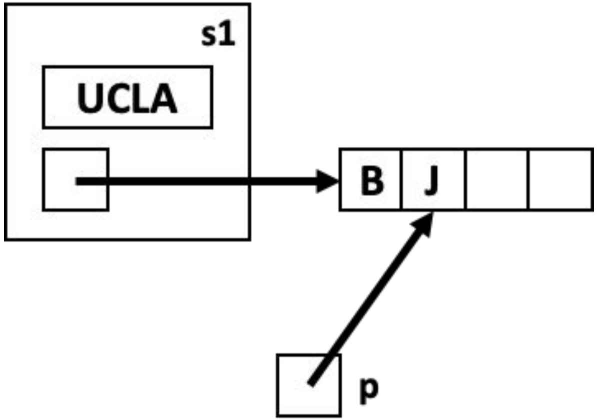
```
#include <iostream>
#include <string>
using namespace std;

class A
{
public:
    A(){cout << "A()" << endl;}
    A(int x){cout << "A(" << x << ")" << endl;}
    ~A(){cout << "~A()" << endl;}
};
class B : public A
{
public:
    B():a1(1),a2(2){cout << "B()" << endl;}
    ~B(){cout << "~B()" << endl;}
private:
    A a2;
    A a1;
};

int main()
{
    B b;
    return 0;
}
```

Copy Constructors

Motivation

Consider an example of UCLA and its students:

```
Student st1(“Brian”);
Student st2(“John”);
School s1(“UCLA”);
s1.addStudent(st1);
S1.addStudent(st2);
Student *p = s1.getStudent(“John”);
```



We want to create a new School called s2, with exactly the same content as s1. In other words, we want to clone s1.

```
School s2(“”); s2=s1; // Is this correct?
```

Copy Constructors

Shallow copy problem

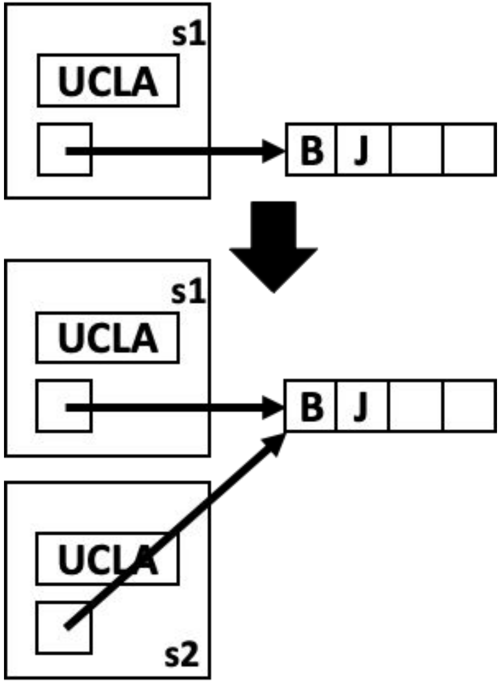
UCLA Samueli Computer Science

We want to create a new School called s2, with exactly the same content as s1. In other words, we want to clone s1.

```
School s2(“”); s2=s1; // Definitely not!
```

What if grab values out of s1 and manually copy them into s2?

```
School s2(“”);
s2.setName(s1.getName( ));
… // copy all members and properties
```

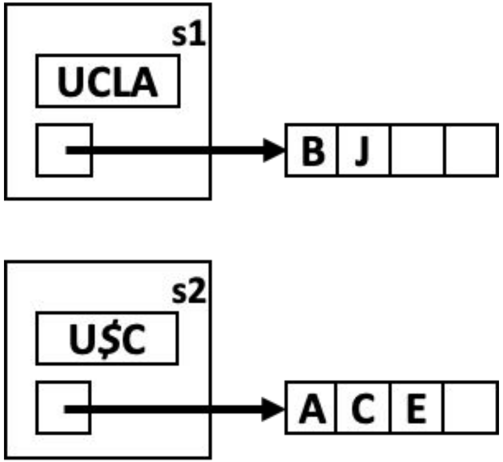


Copy Constructors

Example

Immutable. We don't change the object we're copying from.

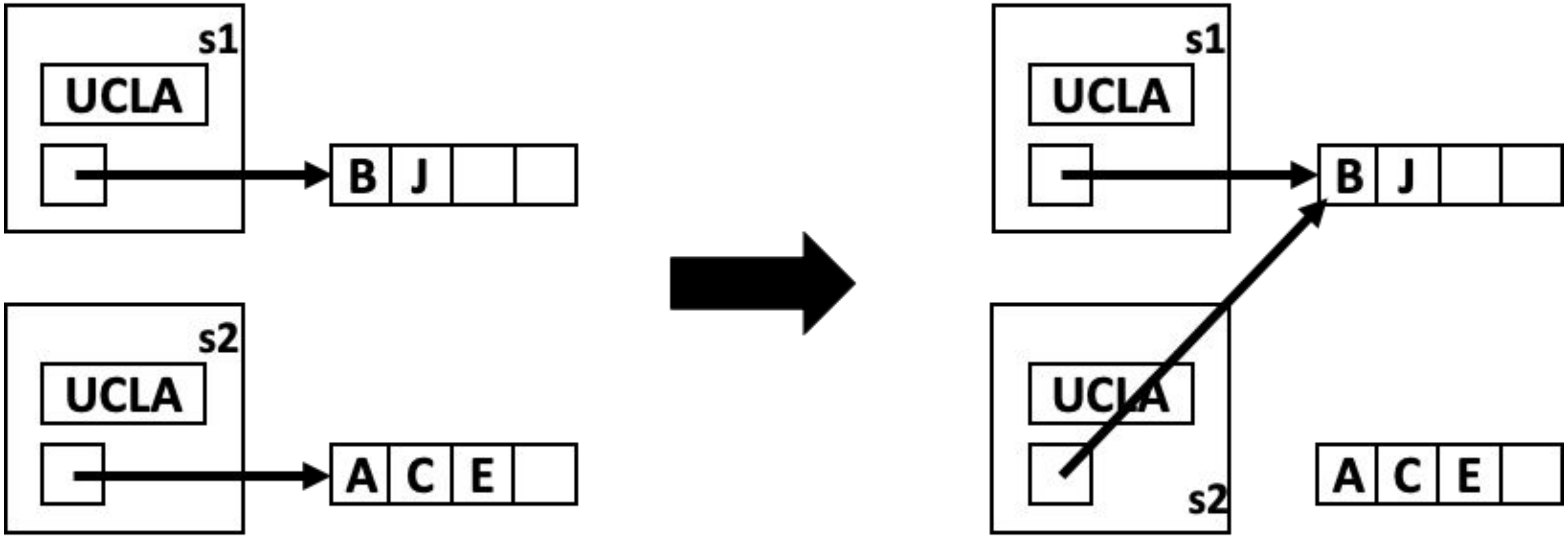
```
School::School(const School &aSchool){
    m_name = aSchool.m_name;
    m_numStudents = aSchool.m_numStudents;
    m_students = new Students[m_numStudents];
    for (int i = 0; i < m_numStudents; ++i)
        m_students[i] = aSchool.m_students[i];
}
```



Assignment Operator

Motivation

What if `s2 = s1`?



Assignment Operator

Example

UCLA Samueli Computer Science

Assignment operator

```
School& School:: operator=(const School &aSchool){
    m_name = aSchool.m_name;
    m_numStudents = aSchool.m_numStudents;
    m_students = new Students[m_numStudents];
    for (int i = 0; i < m_numStudents; i++)
        m_students[i] = aSchool.m_students[i];

    return *this;
}
```

*Do not forget *this!*

Assignment Operator

Example

Make it better:

```
School& School:: operator=(const School &aSchool){
    if (this != &aSchool)
    {
        m_name = aSchool.m_name;
        m_numStudents = aSchool.m_numStudents;
        delete[] m_students;
        m_students = new Students[m_numStudents];
        for (int i = 0; i < m_numStudents; i++)
            m_students[i] = aSchool.m_students[i];
    }
    return *this;
}
```

Data Structure

Before we talk about arrays, let’s talk about data structure first

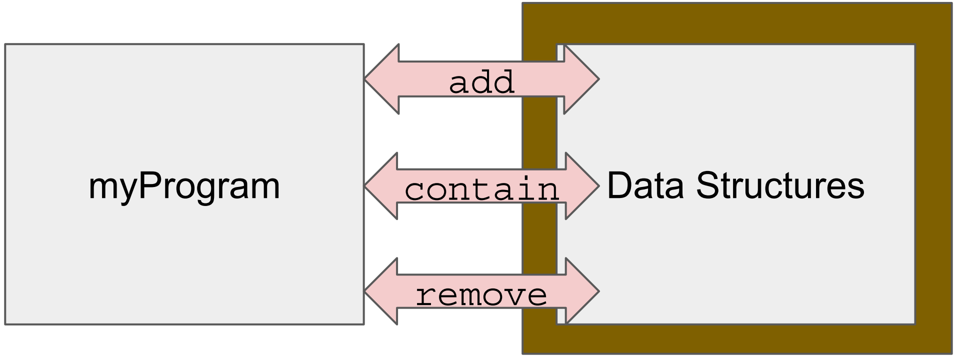
CS32 is not just to write code and run. It is more about organizing data. We call an organization scheme a data structure. For every structure, we define:

- How to store/organize the data items?
- Method to add new data / remove data
- Apply functions on data: most importantly, how to search data?

We also need to know pros and cons of each data structure as well as its efficiency or complexity of algorithms with these data structures.

Arrays

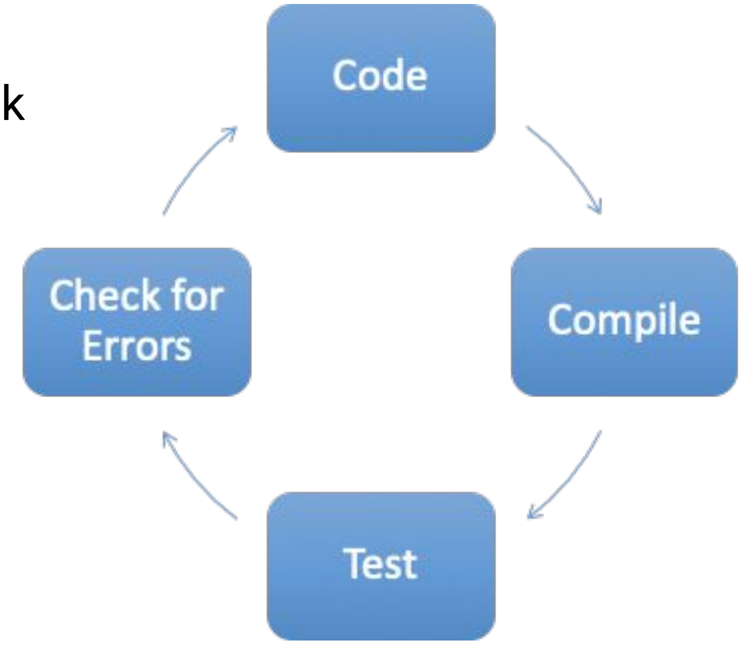
- Most basic data structure and most important!



- Add / Contain / Remove → Complexity?
- More implementations: getIndexOf, getFrequencyOf, …
- More questions: Using array to implement Linked List? How to sort a array or maintain a sorted array?

Some basics

- Pointers → Memory layout
- Assert
- Debug → Breakpoint, trace, variables, stack
- Pseudocode
- Others…



Thank you!

Q & A

Acknowledgement: Muhao Chen, Youfu Li

Group Exercise

Exercise problems from Worksheet 1 (see “LA worksheet” tab in CS32 website).

Answers will be posted after discussions.