CS3157: Advanced Programming

Lecture #14

Apr 17

Shlomo Hershkop

shlomo@cs.columbia.edu

1

Outline

- Wrapping up CPP
  - Little things
  - dynamic memory allocation (new/delete vs malloc/free)
  - Copy and construction options
  - Templates
  - Polymorphism

  - c++core ch 7-9,11-13

2

Announcements

- How are you doing on the homework ?
- Anyone up for extension ??
- Wednesday lab:
  - Due from last week
  - Get it in on time please
  - Will allow you more time to focus on other stuff

3

Linkage directions

- If you want to call a function in another programming language, the compiler must be told that different rules apply
- Linkage directive
  - Single statement
  - Compound form
- Declared outside of functions

4

Single form

- extern “C” void something(int);

- Keyword
- String
- Function
- Compiler will type check any function calls

5

Compound form

- extern “C” {
  int printf(const char * …);
  int scanf (const char * …);
  }
- extern “C” {
  #include <cmath>
  }

6

Other languages

- Depends on the compiler

- For example many support
- FORTRAN

7

Dynamic allocation

- Local variables have local life and scope
- If you want to dynamically create and manage memory, use the new and delete
- Using pointers

- Have to be careful from dangling pointers…
- Ideas?

8

Reality check

- int *p = new int (1024);
- int *q = new int [1024];
- int (*r)[1024] = new int [4][1024];

9

Abstraction and member functions

- How are object internally manipulated by cpp…..lets take a look at a complex example

10

Rect

```
class Rect {

  // ...
  private:
  int top, left;
  int width, height;
  ..
  };
```

11

Color

```
class Color{
  // ..
  private:
  int data;
  };
```

12

TextBox

```
class TextBox: public Rect{
  //...
  private:
  Color txtColor;
  int frameThick;
  char *text;
  };
```

13

main

```
main(){
  TextBox source, dest;

  //...

  dest = source;
```

- How to get this to work ?

14

Overloading operator =

```
class TextBox : public Rect{
  public:
  void operator=(TextBox &source);
  ..
```

15

Equivalent

```
main(){
  TextBox source, dest;

  //...

  dest.operator=(source);
```

16

Inside

```
void TextBox::operator=(TextBox &source) {

    if(this == &source)
        return;

    Rect::operator=(source);

    txtColor = source.txtColor;

    frameThick = source.frameThick;

    delete []text;
    if(source.text != 0) {
      text = new char[strlen(source.text+1)];
      strcpy(text,source.text);
    }
    else
        text = 0;
    }
```

17

Implicit assignment

- If you don’t define an assignment operator
  - Will try to figure out how do to it
  - By looking at each field member variable
  - Works with primitives
  - Pointers will get shallow copied

18

Copy constructor

- TextBox t2 = t1;

- Looks like assignment
- Really a constructor call with object as argument
- Called copy constructor
- Combination of constructor and assignment

19

Defining it

- Just overload the constructor
- TextBox(TextBox &source);

- Be careful:
  - When you overload the copy constructor you throw out a default constructor
  - Which means you need to explicitly define a default constructor (no arg)

20

code

```
TextBox::TextBox(TextBox
 &source){

 Rect::operator=(source);

 frameThick = source.frameThick;
 textColor = source.textColor;

 etc
```

21

Chaining

- If you want to be able to say

```
Textbox a,b,c;
//…
a = b = c ;
```

- how would the operator overloaded be different ??

22

Exception

- Like in java , CPP allows you to throw and catch exceptions

- Compiler time exceptions
- Run time exceptions

23

Template programming

- Allows you to specify a type to pass in to your class, so can create a collection class to handle many different types, without having the problem if limited casting in the code

- Allows you to move errors from run time to compiler time

24

Virtual functions

- Allows you to declare a function in the base class without a definition
- Each of the derived class provide a definition unique to their implementation
- At runtime will allow all derived class object instances to be manipulated uniformally

25

Next week

- Please finish the lab for this Wednesday
- Homework extended till Wednesday night
- Ta’s will be in lab to help with homework

- Read up on things discussed in today’s class
  - Understand how operator overloading works and implications
  - Understand the pointer examples
- Will be starting shell programming next class

26