

Introductory Tutorial for CIS 3700, C++, and STL

Thursday, January 18, 2007

Teaching Assistants: Robert Collier and Wei Wang

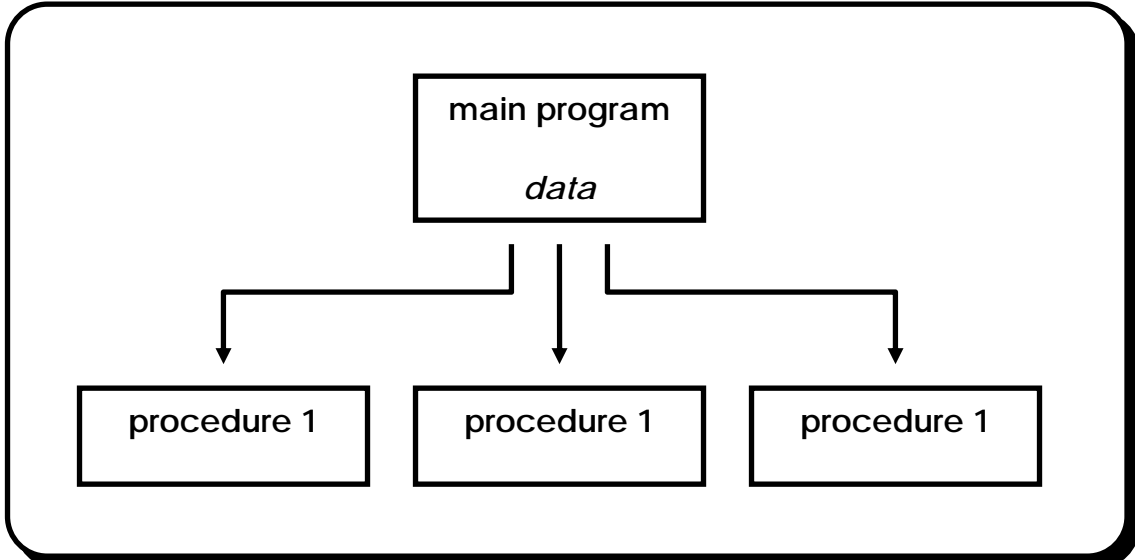
Email: ta3700@cis.uoguelph.ca

Office Hours: Monday 11:30 - 12:30 (Robert)
Friday 11:30 - 12:30 (Wei)
... or by appointment

Tutorials: MacKinnon 223
Thursday 11:30 - 12:30
(Optional)

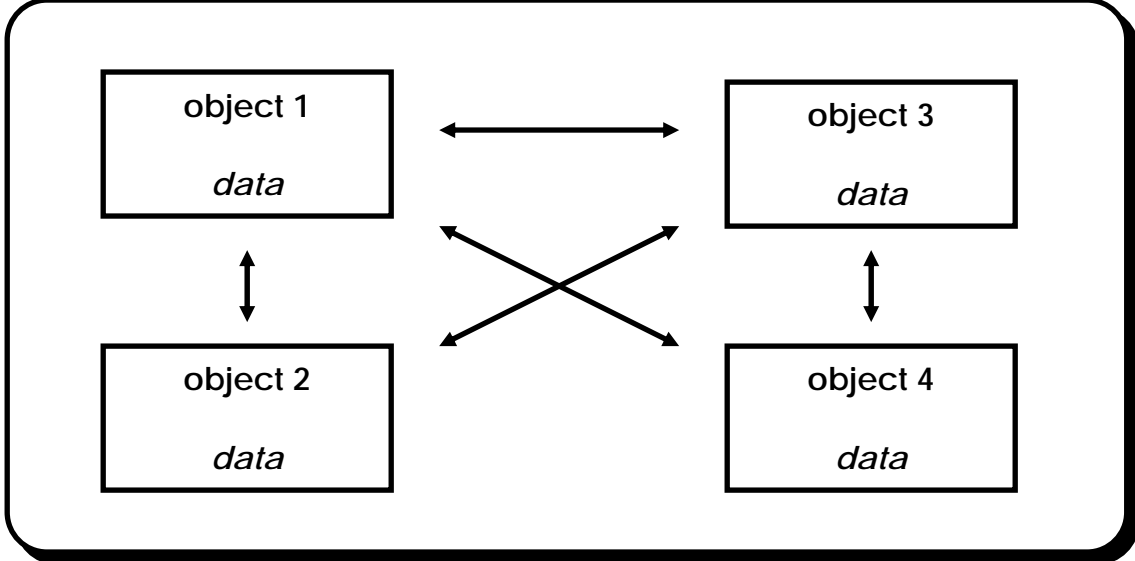
Procedural Programming vrs. Object Oriented Programming

Procedural Programming



"divide programming task into data structures and procedures"

Object Oriented Programming



"divide programming task into objects"

C++ Basics

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

class CRectangle {
private:
    int width, height;
public:
    CRectangle ();
    CRectangle (int,int);
    ~CRectangle () {};
    void set (int,int);
    int area () {return (width*height);}
};

CRectangle::CRectangle () {
    width = 5;
    height = 5;
}

CRectangle::CRectangle (int a, int b) {
    width = a;
    height = b;
}

CRectangle::set (int a, int b) {
    width = a;
    height = b;
}

int main () {
    CRectangle x (3,4);
    CRectangle y;
    cout<<"Area of x: "<<x.area()<<endl;
    cout<<"Area of y: "<<y.area()<<endl;
    return 0;
}
```

Access Specifiers

```
class class_name {  
    access_specifier_1:  
        member1;  
    access_specifier_2:  
        member2;  
    ...  
}
```

private:

accessible from within other members of the same class
(also, from friend classes)

public:

accessible from anywhere the object is visible

Constructors and Destructors

Constructor - **CRectangle (int,int)**
automatically called when new object is created
has same name as the class and has no return type

Default Constructor - **CRectangle ()**
constructor called when no arguments are passed

Destructor - **~CRectangle ()**
automatically called when object is destroyed
(either scope has finished or delete operator used)
has same name as class (with ~) and no return type

C++ Templates

```
void swap (int& a, int& b) {  
    int tmp = a;  
    a = b;  
    b = tmp;  
};
```

...WOULD NEED A SEPARATE METHOD FOR EACH DATA TYPE..

```
template <class anytype>  
void swap (anytype& a, anytype& b) {  
    anytype tmp = a;  
    a = b;  
    b = tmp;  
};
```

...BUT EQUALS IS NOT DEFINED FOR A CRectangle...

```
CRectangle& operator= {const CRectangle& R} {  
    width = R.width;  
    height = R.height;  
    return *this;  
}
```

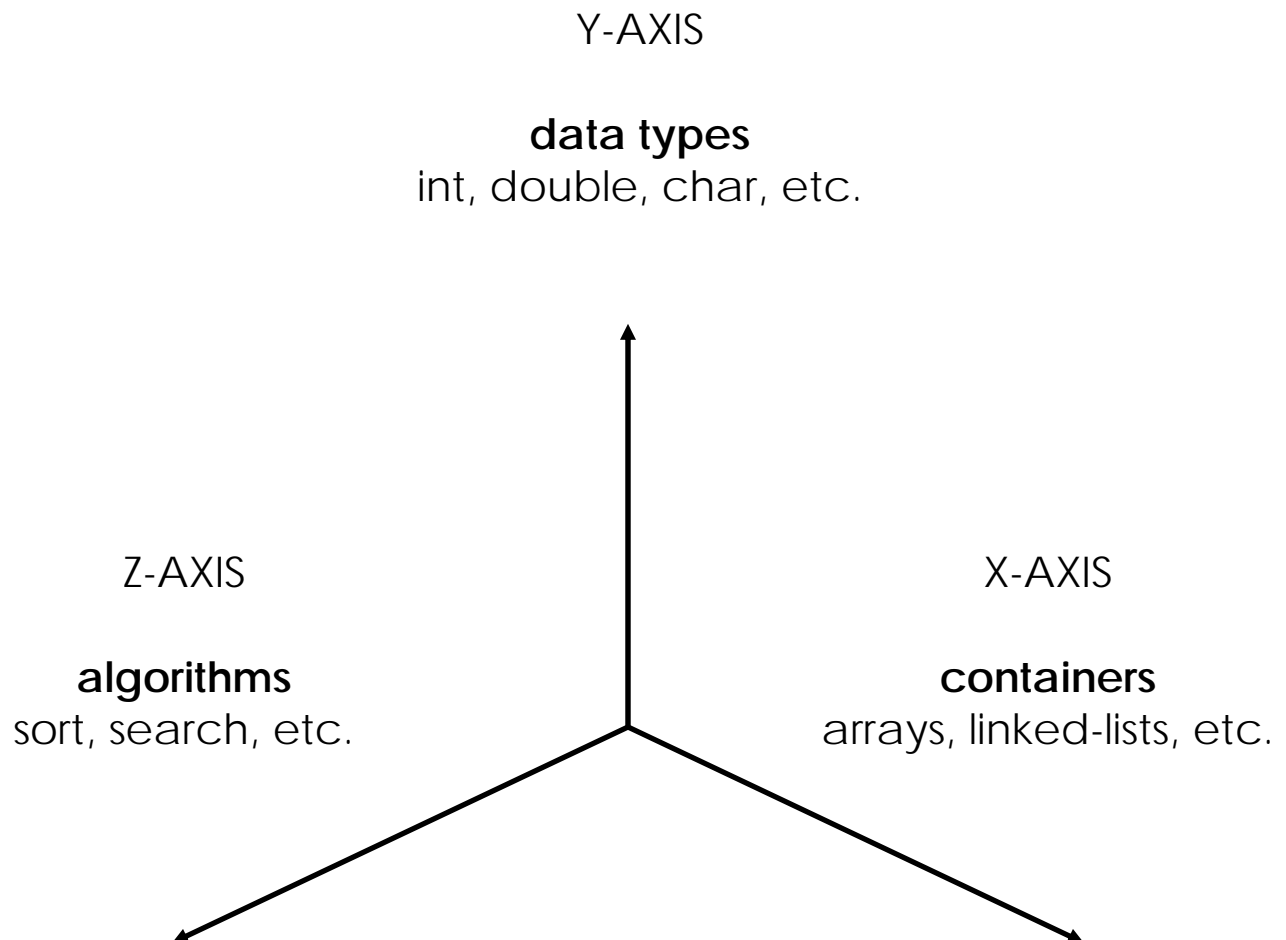
...NOW, THIS IS POSSIBLE...

```
int a = 3, int b = 5;  
CRectangle x;  
CRectangle y;  
swap(a, b);  
swap(x, y);
```

Standard Template Library (STL)

- C++ library of containers, algorithms, and iterators
- provides many of basic algorithms and data structures

THE CONCEPT BEHIND IT ...



... TEMPLATES TAKE CARE OF THE Y-AXIS ...

... STL HELPS REDUCE THE X-AXIS ...

Simple STL Example - Vectors

Constructors

```
vector <anytype> v;
```

creates an empty vector of anytype

```
vector <anytype> v(n);
```

creates a vector with n values

Functions

```
v.size(); // returns int
```

number of elements in vector v

```
v.capacity(); // returns int
```

(current) maximum number of elements

```
v.empty(); // returns bool
```

true if (v.size() == 0), false otherwise

```
v.push_back(e);
```

adds e to the end of v

```
v.pop_back();
```

removes last element of v

```
v.clear();
```

remove all elements of v

Simple STL Example - Vectors

More Functions

`v.at(i);` (also, `v[i]`)
access or set the *i*th element of *v*

`v.front();`
access or set the first element

`v.back();`
access or set the last element

Iterators (...more on these in the next tutorial...)

`v.begin();`
returns an iterator to the first element

`v.end();`
returns an iterator to the last element

`v.insert(iterator, e);`
inserts *e* at the position of the iterator

Trivial Sample Source

```
vector<int> v(3);  
v[0] = 7;  
v[1] = v[0] + 3;  
v[2] = v[0] + v[1];           // 7, 10, 17  
reverse(v.begin(), v.end()); // 17, 10, 7
```