CS342: Organization of Prog. Languages

Topic 9: [Language] An Overview of C++

- C with methods
- Constructors and Destructors
- Classes
- Inheritance
- Templates
Lesson 1. C with methods

- C++ adds the idea of methods onto C.
- Methods are fields of a structure which act like functions and can access the other fields.
- The names of structs are automatically introduced as typedefs.
- Structures with methods are called *objects*. 
• Method example:

```c
struct Rectangle {
    int width, height;
    int area() {
        return width * height;
    }
    void setSize(int w, int h) {
        width = w;
        height = h;
    }
};

#include <stdio.h>
int main() {
    Rectangle r;
    r.setSize(10, 20);
    printf("%d\n", r.area());
    return 0;
}
```
Lesson 2. Constructors and Destructors

• If a function with the name of the type is provided, it is used to initialize declared variables.

• It is also used to initialized objects allocated with `new`.

• Another special function, with the name of the type preceded by tilde, is used to finalize an object as it is de-allocated.

• This is applied both to local objects on the stack, and to allocated objects de-allocated with `delete`.
• Constructor/Destructor example:

```c
#include <stdio.h>

struct Rectangle {
  int width, height;
  int area() { return width * height; }

  Rectangle(int w, int h) {
    width = w;
    height = h;
  }

  ~Rectangle() {
    printf("%dx%d going away...\n", width, height);
  }
};

int main() {
  Rectangle r(10, 20), *prect;
  printf("Area %d\n", r.area());

  prect = new Rectangle(11, 13);
  printf("Area %d\n", prect->area());
  delete prect;

  return 0;
}
```
Output is...

Area 200
Area 143
11x13 going away...
10x20 going away...
Lesson 3. Classes

- A class is a structure which has some of its fields declared private and others public.

- Class example:

```c
#include <stdio.h>

class Rectangle {
private:
    int width, height;
public:
    Rectangle(int w, int h) { width = w; height = h; }
    int area() { return width * height; }
};

int main() {
    Rectangle r(10, 20), *prect;
    printf("Area %d\n", r.area());

    prect = new Rectangle(11, 13);
    printf("Area %d\n", prect->area());
    delete prect;

    return 0;
}
```
Lesson 4. Inheritance

- A class may be defined as an extension of another.
- We say the second class is *derived* from the first.
- We also say that the first class is a *base class*.
- A base class part is initialized with a call to its constructor following a “::” and before the function body.
Inheritance example:

```c
#include <stdio.h>

class Polygon {
private:
    int nsides;
public:
    Polygon(int n) { nsides = n; }
    int sideCount() { return nsides; }
};

class Rectangle : public Polygon {
private:
    int width, height;
public:
    Rectangle(int w, int h) : Polygon(4) {
        width = w;
        height = h;
    }
    int area() { return width * height; }
    ~Rectangle() {
        printf("%dx%d going away...\n",
            width, height);
    }
};
```
int main() {
    Rectangle r(10, 20), *prect;
    printf("Area %d\n", r.area());

    prect = new Rectangle(11, 13);
    printf("Area %d\n", prect->area());
    delete prect;

    return 0;
}
Lesson 5. Templates

- Templates introduce a parametrized family of functions or classes.
- By filling in the blanks, you get the functions or classes you need.
- Template definitions have to be included in the files where they are used.
- The parameters are given in angle brackets: "<" ">".

• Template example:

```c++
#include <stdio.h>

template<class Length>
class Rectangle {
private:
    Length width, height;
public:
    Rectangle(Length w, Length h) { width = w; height = h; }
    Length area() { return width * height; }
};

int main() {
    Rectangle<double> r(10.5, 20.5);
    Rectangle<int> *prect;

    printf("Area %g\n", r.area());
    prect = new Rectangle<int>(11, 13);
    printf("Area %d\n", prect->area());
    delete prect;

    return 0;
}
```