12.010 Computational Methods of Scientific Programming

Lecturers

Thomas A Herring
Chris Hill
Summary

• Today we finish up C and start C++
• Final C topics
  – Structures: A method for grouping like variables together
  – Memory management
• Start of C++
  – History
  – Ideas of classes and objects
  – Examples to demonstrate concepts
Structures and Types

• Struct alone is still unclear - typedef
  
typedef struct { double cx;
        double cy;
        double cz; } t_point;

main() {
    t_point point;
    point.cx = 3.; point.cy=3.; point.cz=2.;
    plot(point);
}
Structures and Types

- Derived types just like basic types
  - e.g. can use arrays
- typedef struct {
  double cx;
  double cy;
  double cz;
} t_point;

```c
main() {
  t_point point[10]; int i;
  for (i=0; i<10; ++i) {
    point[i].cx = 3.; point[i].cy = 3.; point[i].cz = (double)i;
  }
  for (i=0; i<10; ++i) {
    plot(point[i]);
  }
}
```
Memory Management

- Application code creates variables and arrays at runtime
- `<stdlib.h>` - `malloc`, `calloc`, `free`, `realloc` + `sizeof`
  - e.g

```c
main(int argc, char *argv[]) {
    double *foo; int nel; int i;
    /* Create an array of size nel at runtime */
    sscanf(argv[1], "%d\n", &nel);
    foo = (double *) calloc(nel, sizeof(*foo));
    if ( foo == NULL ) exit(-1);
    for (i=0; i<nel; ++i) { foo[i]=i; }
    free(foo);
}
```
Remember - *, &

short a; short *ptr_to_a;
a = 1;
ptr_to_a = &a;
*ptr_to_a = 1;

Here compiler allocated memory for you

Here application allocates memory explicitly.
Allows more control but requires careful bookkeeping.
Towards C++

• C essentials
  – syntax v. fortran
  – call by reference v. call by value
  – pointers
  – structure, typedef
  – memory management

• C is also the basis for C++
C++

- Object Oriented - Allows you to build/compose very complex applications from building blocks
- Appeared around 1984 (Bjarne Stroustrup, Bell Labs)
- ANSI standard 1997
- Syntax is like C. Getting started: a few extra keywords + few new formalized concepts.
- Book “C++ The Core Language” – O’Reilly
- Successful because you can compose applications from other peoples building blocks. Windows etc.…
- Very complex in detail, like Mathematica takes many years to learn everything!!
C++ concept

• C language + classes
• Class is a formal way to think about good program design.
  – Modularity, encapsulation, hierarchy, abstraction
• A class has
  – Methods (program logic)
  – Data (variables)
  – can be private or public
• Example “string”
  – Methods: set, get
  – Data: string text, string length
```c++
main()
{
    String s;
    printf("Executable code starting\n");
    s.set("Hello");
    printf("%s\n", s.get());
    printf("Executable code ending\n");
}
Compile using g++
Will write out hello + some other stuff
```
C++ Basic Example

```c
main()
{
    String s;
    printf("Executable code starting\n");
    s.set("Hello");
    printf("%s\n", s.get());
    printf("Executable code ending\n");
}
```

- **Class**
- **Instance of the Class**
- **Methods**
String Class - Behind the Scenes

/* ===== Class interface definition ===== */
class String {
public:
  String(); /* Constructor */
  ~String(); /* Destructor */
  void set(char *s); /* Set a string */
  char *get(); /* Get string value */
private:
  char *str; /* Pointer to the string */
  int lngth; /* Length of the string */
};
String Class – Example Methods

/* Set str to point to a private copy of s */
void String::set(char *s) {
    lngth = strlen(s);
    str  = new char[lngth + 1];
    strcpy(str, s);
}

/* Return the pointer to the string */
char *String::get() {
    return str;
}
String Class – Example Methods

/* Constructor */
String::String() {
    str = 0;
    set("");
    printf("I created a string\n");
}

/* Destructor */
String::~String() {
    delete[] str;
    printf("I deleted a string\n");
}
Application Example

Throwing a ball in the air

Get initial velocity and length of “experiment”.

Calculate time evolution of $w$ and $z$.

Print out “trajectory”

$$w = w_0 + gt$$

$$z = z_0 + \int_0^t w dt$$
C “Procedural” Form

main ( )
{ float t=10.; float w0=10.;
  t_gball *theBall;/* Stats for the ball */

  /* Allocate space for full ball time history */
  createBall(w0, &theBall );
  /* Step forward the ball state */
  stepForwardState( t, &theBall );
  /* Write table of output */
  printTrajectory( t, w0, theBall);
}
C++ Using “Ball” Class

main()
{float w0 = 10.; float t=10.;
 Ball b;
 b.initialize(w0);
 b.simulate(t);
 b.printTrajectory();
}

All info. is held in “b”. Fewer args, cleaner “abstraction”.
Summary

• Finished up C with structures and memory management
• Started with C++
  – C++ is C with the addition of “classes”
  – Class is a formal way to think about good program design.
    • Modularity, encapsulation, hierarchy, abstraction
  – A class has
    • Methods (program logic)
    • Data (variables)
    • can be private or public