CSE 459.22
Programming in C++
Neelam Soundarajan
Computer Sc. and Engineering
Dreese Lab 579
Tel: 292 1444

Please send corrections by e-mail to neelam@cse

Who I am:
Faculty member in the CSE Dept.
I usually teach CSE 655, 755.

Who you are (or should be):
CIS/CSE major who has completed CSE 321.

Others:
Must have a reasonable background in programming/software. This is not a beginning course in programming.

Course focus:
Important technical details of C++ including some “low-level” details.

Focus will be on the core language, not details such as differences between various implementations etc.

Some Terminology:
Procedural programming: Focus is on the algorithms necessary to do the required computation. Data structures introduced as needed.

Modular programming: The program is partitioned so that each set of procedures and the data they manipulate is grouped into a separate module. A given module may be changed without affecting the correctness of the other modules.

Terminology (contd.):
ADT-based programming: Often, multiple instances of a particular kind of data are needed. So: create corresponding types. The rest of the program uses the types but doesn’t know the internal details.

Object-oriented programming: Somewhat similar but often the “objects” correspond to entities in the actual (real-world) system. Plus, some important technical differences.

In most languages (including C++) both ADT-based and OOP are implemented using classes.

Reading: Chapter 2.
Types (Ch. 4):

Basic types:

- bool, char, int, double
- (and slightly fancier versions)
- enum ("enumerated" type).

Pointer types:
- int*

Array types:

Types and declarations (contd.):

- Before a name is used it must be declared.
- char ch;
- string s; // Not a "built-in" type
- int count = 1;
- const double pi = 3.14;
- enum Months { Jan, Feb, Mar, ...};
- int lab1Grades[40];
- extern int errorNo; // Declaration
- const char* name = "C++";
- struct Date { int d, m, y};
- int date(Date* p) { return p->d; }
- double sqrt(double); // Declaration
- typedef int MyInt;
- const c = 7; // Error! no type specified.
- gt(int a, int b) { return (a>b) ? a : b; }
- // error: return type?

Scope:

- int x; // global x;
- void f() {
-   int x; // local x; hides global x
-   x = 1; // assign to local x;
-   {
-     int x; // hides first local x;
-     x = 2; // assigns to this x;
-   }
-   x = 3; // assigns to first local x
- }
- int* p = &x; // addr. of global x

Exercise: Read Section 4.9.5 (Initialization).

Exercise: Do problem (1), Section 4.11 (p. 85).
Pointers, Arrays, Structures (Ch. 5):

For any type \( T \), \( T* \) is the "pointer to \( T \) type".

\[
\begin{align*}
\text{int* pi; // pointer to int} \\
\text{char** ppc; // pointer to pointer to char} \\
\text{int* ap[15]; // array of 15 pointers to ints} \\
\text{int (*fp)(char*); // ptr. to fn. taking a char* arg;} \\
\text{int* f(char*); // fn. taking a char* arg., returning ptr. to int} \\
\end{align*}
\]

\[
\begin{align*}
\text{char c = 'a';} \\
\text{char* p = &c; // p holds addr. of c} \\
\text{char c2 = *p; // c2 == 'a'} \\
\end{align*}
\]

\[
\begin{align*}
\text{void* vp; // can point to ANY type} \\
\end{align*}
\]

Pointers, Arrays, Structures (contd.):

For any type \( T \), \( T[\text{size}] \) is the "array of \( \text{size} \) elements of type \( T \)", indexed from 0 to \( \text{size}-1 \).

\[
\begin{align*}
\text{double v[3]; // array of three doubles, v[0], v[1], v[2]} \\
\text{char* a[32]; // array of 32 ptrs t char: a[0], \ldots a[31]} \\
\end{align*}
\]

\[
\begin{align*}
\text{void f(int a) {} } \\
\text{\quad int v1[i]; // illegal; size not a constant exp.} \\
\text{\quad vector<int> v2(i); // ok; uses STL} \\
\text{\}} \\
\text{int d2[10][20]; //array of 10 arrays of 20 ints} \\
\text{int d3[10,20]; // error!} \\
\end{align*}
\]

**Exercise:** Read Section 5.2.1 (initialization).