Structure of Programs

Computer programs are among the most complex artifacts engineered by humankind. Consider the following:

1. A 700 page novel occupies 2.6 MB of disk space.
2. Windows XP takes about 1 GB.
3. A switch built by Nortel in the mid 90's took about 6 million lines of code.

How do we manage such complexity? We use a strategy known as **Divide and Conquer**, one of the oldest strategies used by humankind.

1. Complex systems are **modularized**—broken into smaller more manageable pieces.
2. The modules themselves are often broken into smaller modules, and those into smaller modules again.
3. The set of modules is organized **hierarchically** to constitute the system.

This schema is widely used.

**Structure of the Roman Army—a rigid hierarchy**

With 6000 men under his command a Roman Legate couldn't know every legionary. His job was to worry about the big picture. He would give commands like, "Move the 1st cohort forward" or, "Hold the 3rd cohort in reserve."

At the bottom level, the centurion would know each of his legionaries. He would decide who best should be on the critical left edge of the shield wall, who in the centre. It was his job to sweat the details.

**Modern Corporations—A Flexible Hierarchy**

- The Atlantic Group
- Rhino Records
- Elektra Entertainment Group
- London-Sire Records Inc
- Warner Bros. Records
- Warner Music International
- Time Life Music
- Columbia House (joint venture)
- Giant (Revolution) Records (joint venture)
- Maverick (joint venture)
- Qwest Records (joint venture)
- Ruffnelion Records (joint venture)
- Sush Pop Records (joint venture)
- Tommy Boy Records (joint venture)

Production/Distribution
- Warner Brothers Studios
- Castle Rock Entertainment
- New Line Cinema
- Fine Line Features

Internet
- AOL
- CompuServe
- Netscape
- AOL Mousefire
- Digital City
- MacQuest
- Spinrique

Film
- AOL Time Warner
  - The Atlantic Group
  - Rhino Records
  - Elektra Entertainment Group
  - London-Sire Records Inc
  - Warner Bros. Records
  - Warner Music International
  - Time Life Music
  - Columbia House (joint venture)
  - Giant (Revolution) Records (joint venture)
  - Maverick (joint venture)
  - Qwest Records (joint venture)
  - Ruffnelion Records (joint venture)
  - Sush Pop Records (joint venture)
  - Tommy Boy Records (joint venture)

Sports
- Atlanta Braves
- Atlanta Hawks
- Atlanta Thrashers
- Turner Sports

Recreation
- World Championship
- Wrestling
- Goodwill Games

Magazines
- Time Magazine
- Life Magazine
- Fortune Magazine
- Sports Illustrated
- Money
- People
- Entertainment Weekly

Production/Distribution
- HBO Independent Productions
- New Line Television
- Turner Original Productions
- Warner Brothers Television
- Warner Brothers Animation
- Looney Tunes

http://www.engr.mun.ca/~mpbl/teaching/2420/lectures/introduction/program_structure.htm  2004.01.14
Straightening out the Hierarchy

1. Each corporation creates its own hierarchy, according to its own needs
2. Hierarchies tend to grow organically
3. Nevertheless, chart typical of modern mega-corporation

Common Features of Hierarchies

1. Drawing shows modules for a modern army—used by many countries
2. Really a tree since an army has a number of divisions, a division a number of battalions and so on
3. Different kinds of modules at different levels.
4. Modules towards top are bigger, more complex
5. Modules towards bottom are smaller, simpler
6. Modules towards top have high overhead—generals

Each company has its own hierarchy

High degree of complexity
Early in 20th century, Americans introduced the squad
Smallest unit of all
Handfull of people under a single corporal (very low overhead)

Very focussed—one job at a time
- Hold south end of the bridge
- Scout that village
- Take that machine gun

In this course we will focus on the squads of computer programming.
1. small amounts of code
2. single well specified task
3. minimal overhead

Programming Modules
Computer programs modularized

Classes (file)
Menu
Window
ScrollBar
Functions
close()
minimize()
open()

Teaching Machine
Typical medium scale program
1. 7000 programming hours
2. 14 packages
3. 14 sub-packages
4. over 700 classes
5. around 5000 functions

Mini-Programs

Full model too complex to teach initially

Concentrate on a mini-
An Even Smaller Model

We can reduce the model above even farther by only having the main function

This is what we will start with.

It is important to remember that what we are actually learning how to do is to write a single function

Writing a program with more than one function is not much different

Notice that the instructions above are numbered.

This is to indicate that instructions in a program are executed in sequence.

1. The first instruction is executed
2. Then the second instruction is executed
3. Then the third instruction is executed

and so on.

The Build Process

Computer programs have to be built. We use a number of processes (computer programs) to build a program
An **editor** is a specialised word processor used to prepare source modules in the language of choice (e.g. C++, Java, Fortran, Basic).

The **precompiler** adds in standard pre-written code (boilerplate) from include files you specify to produce a complete source module.

The **compiler** produces object code for the target computer/operating system.

The precompiler is like a secretary that helps you pull together a full source document.

The compiler is like a translator that converts your module from the language of your choice (C++) to language the computer (PC, MAC, Sun) understands.

---

**A Traditional First Program**

Here is a traditional first program you will see in almost every textbook in C++.

```c++
#include <iostream>
using namespace std;
int main(){
    cout << "Hello world!";
    return 0;
}
```

The first line is an instruction to our precompiler "secretary" saying we want to use the standard input/output system (the iostream). The secretary (the precompiler program) will go and fetch all the boilerplate code from the iostream file and insert it right where we wrote the line.

The second line makes reference to where the names we’re going to use may be found. It’s a little bit like the statement you might find in the beginning of some engineering textbooks saying *in this book we’re going to use the ISO names*.

The third line is blank, a separator to help our eye group logical entities together.

An **executable** is a program that will run on the computer. The editor, precompiler, compiler and linker are all **executables**.

So is your program!
The fourth line is the beginning of our one and only function, `main`. Every program has one (and one only) `main` function. So if your program has only one module, it's got to be `main`.

The fifth line is our first line of active computer code. It is an instruction to the computer saying, *please output the words, “hello world!”*

We'll explain the sixth line later.

This page last updated on Thursday, January 8, 2004