

Software Techniques

Tim Ji
Lecture 1

Outline

1. Course Information
2. Overview of C Programming
3. My first program : Hello, World!

Lab 1

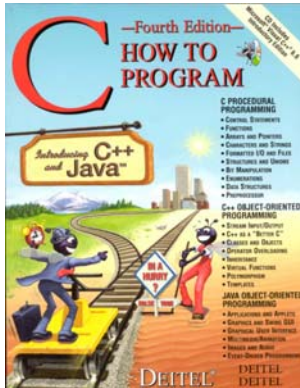
Visual C++ 6.0 IDE

Readings

Deitel Chapter 1, 2

1.1 Course Information

- **Official Course Outline**
 - On course webpage, MS Word document
 - Available in the first lecture
- **Course Objectives/Learning Outcomes**
 - Write structured code in a subset of C language
 - Be able to design, implement, debug, and test programs
 - Be able to analyze programs
- **Textbook: Deitel, “C How to Program”**
 - This book also covers C++ and Java which will not be covered in this course:
 - C++
 - Chapter 15 – 23 introduce the C++ programming language
 - Java
 - Chapters 24 – 30 introduce the Java programming language



Course Information (2)

- **Instructor**
 - Name: Tim Ji
 - Email tji AT conestogac.on.ca
 - Office: 2A503, x2230
 - Office hour: Drop by my office or by appointment
- **Grading**
 - Midterm exam: 20%
 - Final exam: 30%
 - Assignments + Labs: 50%
- **Assignments**
 - Due in class at the end of lecture, 20% reduction in the assignment mark for each school day it is late.
 - The assignments must be your own work. Cheating/copying other people's files will result in both papers receiving zero.

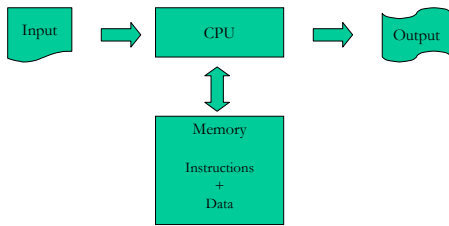
Course Information (3)

- **Labs**
 - Lab tasks will be demonstrated in the following week lab, and lab reports must be submitted by the end of the lab. You have 1 week to finish the lab task.
- **Course website**
 - http://www.conestogac.on.ca/~tji/sw_tech
 - Announcements, assignments, lecture handouts, etc.
 - Check it out at least once per day.
- **Study to pass**
 - You have to pass both the tests portion and the assignments portion
- **Assignments/Labs marking**
 - Demonstrate your assignment in lab
 - Answer questions regarding to your own source code
 - Programming style of printouts of your source code

1.2 Overview of C Programming

- **Computer**
 - Device capable of performing computations and making logical decisions
 - Computers process data under the control of sets of instructions called computer programs
- **Hardware**
 - Various devices comprising a computer
 - Keyboard, screen, mouse, disks, memory, CD-ROM, and processing units
- **Software**
 - Programs that run on a computer

Basic Computer Model



Von Neumann Architecture

Programming Languages

Three types of programming languages

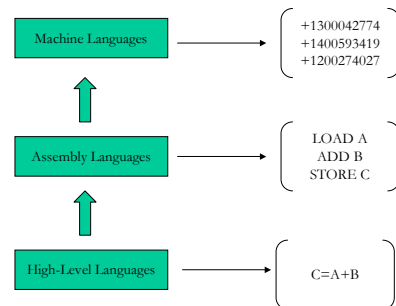
1. Machine languages
 - Strings of numbers giving machine specific instructions
 - Example:
+1300042774
+1400593419
+1200274027
2. Assembly languages
 - English-like abbreviations representing elementary computer operations (translated via assemblers)
 - Example:
LOAD BASEPAY
ADD OVERPAY
STORE GROSSPAY

Programming Languages (2)

Three types of programming languages (continued)

3. High-level languages
 - Codes similar to everyday English
 - Use mathematical notations (translated via compilers)
 - Example:
`grossPay = basePay + overTimePay`

Programming Languages (3)



History of C

- C
 - Evolved by Ritchie from two previous programming languages, BCPL and B
 - Used to develop UNIX
 - Used to write modern operating systems
 - Hardware independent (portable)
 - By late 1970's C had evolved to "Traditional C"
- Standardization
 - Many slight variations of C existed, and were incompatible
 - Committee formed to create a "unambiguous, machine-independent" definition
 - Standard created in 1989, updated in 1999

The C Standard Library

- C programs consist of pieces/modules called functions
 - A programmer can create his own functions
 - Advantage: the programmer knows exactly how it works
 - Disadvantage: time consuming
 - Programmers will often use the C library functions
 - Use these as building blocks
 - Avoid re-inventing the wheel
 - If a premade function exists, generally best to use it rather than write your own
 - Library functions carefully written, efficient, and portable

The Key Software Trend: Object Technology

- **Objects**
 - Reusable software components that model items in the real world
 - Meaningful software units
 - Date objects, time objects, paycheck objects, invoice objects, audio objects, video objects, file objects, record objects, etc.
 - Any noun can be represented as an object
 - Very reusable
 - More understandable, better organized, and easier to maintain than procedural programming
 - Favor modularity

C++ and C++ How to Program

- **C++**
 - Superset of C developed by Bjarne Stroustrup at Bell Labs
 - "Spruces up" C, and provides object-oriented capabilities
 - Object-oriented design very powerful
 - 10 to 100 fold increase in productivity
 - Dominant language in industry and academia
- **Learning C++**
 - Because C++ includes C, some feel it is best to master C, then learn C++
 - Starting in Chapter 15, we begin our introduction to C++

Java and Java How to Program

- **Java is used to**
 - Create Web pages with dynamic and interactive content
 - Develop large-scale enterprise applications
 - Enhance the functionality of Web servers
 - Provide applications for consumer devices (such as cell phones, pagers and personal digital assistants)
- **Java How to Program**
 - Closely followed the development of Java by Sun
 - Teaches first-year programming students the essentials of graphics, images, animation, audio, video, database, networking, multithreading and collaborative computing

Other High-level Languages

- **Other high-level languages**
 - FORTRAN
 - Used for scientific and engineering applications
 - COBOL
 - Used to manipulate large amounts of data
 - Pascal
 - Intended for academic use

Structured Programming

- **Structured programming**
 - Disciplined approach to writing programs
 - Clear, easy to test and debug and easy to modify
- **Multitasking**
 - Specifying that many activities run in parallel

Basics of a Typical C Program Development Environment

- **Phases of C++ Programs:**

1. *Edit*

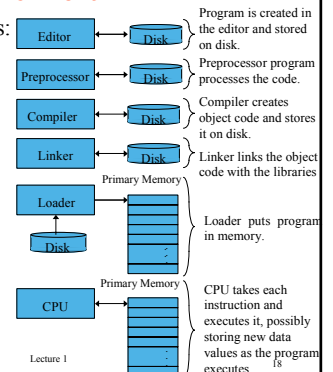
2. *Preprocess*

3. *Compile*

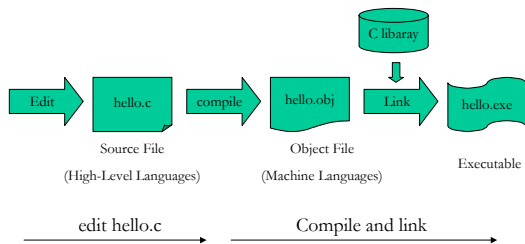
4. *Link*

5. *Load*

6. *Execute*



Creating Programs



My First C Program: Hello World!

```
/* My first C program */
```

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

```
int main()
{
    printf("Hello World\n");
    return 0;
}
```

Comments

```
/* My first C program */
```

- Comments are a way of explaining what a program does.
- They are put after // or between /* */.
- Comments are ignored by the compiler and are used by you and other people to understand your code.
- You should always put a comment at the top of a program that tells you what the program does because one day if you come back and look at a program you might not be able to understand what it does but the comment will tell you.
- You can also use comments in between your code to explain a piece of code that is very complex.
- Here is an example of how to comment the Hello World program:

Updated Comments

```
/* Author: Tim Ji
   Date: 1/10/2005
   Description: Writes the words "Hello World"
   on the screen
  */
```

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

```
int main()
{
    printf("Hello World\n"); //prints "Hello World"
    return 0;
}
```

Preprocessing

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

- This includes a file called stdio.h which lets us use certain commands.
- stdio is short for Standard Input/Output which means it has commands for input like reading from the keyboard and output like printing things on the screen.

Program Entry Point

```
int main()
```

- **int** is what is called the return value which will be explained later on in this course.
- **main** is the name of the point where the program starts and the brackets are there for a reason that you will learn in the future but they have to be there.

Blocks

{

- The 2 curly brackets are used to group all the commands together so it is known that the commands belong to main.
- These curly brackets are used very often in C to group things together.

Functions

`printf("Hello World\n");`

- This is the **printf** command and it prints text on the screen.
- The data that is to be printed is put inside brackets. You will also notice that the words are inside quotation marks because they are what is called a string.
- Each letter is called a **character** and a series of characters that is grouped together is called a **string**.
- Strings must always be put between quotation marks. The `\n` is called an escape sequence and represents a newline character and is used because when you press ENTER it doesn't insert a new line character but instead takes you onto the next line in the text editor.
- You have to put a semi-colon after every command to show that it is the end of the command.

Return from Function

`return 0;`

- The int in `int main()` is short for **integer** which is another word for number.
- We need to use the return command to return the value 0 to the operating system to tell it that there were no errors while the program was running.
- Notice that it is a command so it also has to have a semi-colon after it.

Programming Style

- Indentation

- You will see that the **printf** and **return** commands have been **indented** or moved away from the left side.
- This is used to make the code more readable.
- It seems like a stupid thing to do because it just wastes time but when you start writing longer, more complex programs, you will understand why indentation is needed.

Lab 1: VC++ 6.0 IDE

- Introduction to Visual Studio 6.0
- Lab material is available on the course webpage
- Follow the lab task specifications -- your marks depend on it.

Readings

- Deitel, Chapter 1, 2