Computing and compilers

Comp Sci 1570 Introduction to C++

Outline

1 Objectives
2 Introduction
3 Hardware
4 Software
Low level
High level
Interpreted and compiled
C++
Compilers
Editors
Debuggers
Profilers
Overview
5 Computational thinking
Algorithms
Pseudocode
Basic elements

Objectives

- Evaluate the difference between hardware and software
- Find out about the various types of software
- Get a high level understanding of how program code is transformed into working software applications

Outline

What is a computer program?

- A sequence of instructions that dictate the flow of electrical impulses in a computer system.
- Can perform useful tasks, solve high-level problems, play games.
- We can be oblivious to the lower-level activity
- Most programmers write software at this higher, more abstract level also.
- An accomplished computer programmer can develop sophisticated software with little or no interest or knowledge of the actual computer system upon which it runs.
- Software construction tools hide the lower-level details from programmers, allowing them to solve problems in higher-level terms.

What is a computer system?

- Hardware (only sometimes important to CS)
- Software (CS!)

Outline

Objectives
Introduction
Hardware
Software
Low level
High level
Interpreted and compiled
C++
Compilers
Editors
Debuggers
Profilers
Overview
Computational thinking
Algorithms
Pseudocode
Basic elements

Hardware

- **Peripheral devices:** Input/Output devices, Printers, Monitor, Speakers, Mic, The power supply,
- **Internals:** The "motherboard" with memory chips, The CPU (central processing unit), Hard drive, A lot of wires, etc
- What is a CPU?

What is a transistor?

Outline

1 Objectives
2 Introduction
3 Hardware
4 Software
Low level
High level
Interpreted and compiled
C++
Compilers
Editors
Debuggers
Profilers
Overview
5 Computational thinking
Algorithms
Pseudocode
Basic elements

What is software?

Outline

1 Objectives

2 Introduction

3 Hardware

4 Software
- Low level
- High level
- Interpreted and compiled
- C++
- Compilers
- Editors
- Debuggers
- Profilers
- Overview

5 Computational thinking
- Algorithms
- Pseudocode
  - Basic elements

Machine code

For example, adding the registers 1 and 2 and placing the result in register 6 is encoded:

```
[  op  |  rs |  rt |  rd |shamt| funct]
    0     1     2     6     0     32     decimal
 000000 00001 00010 00110 00000 100000   binary
```

- Computers understand language that consists of sets of instructions made of ones and zeros.
- This computer language is often called machine code.
- Programming a computer directly in machine language using only ones and zeros is very tedious and error prone.

Outline

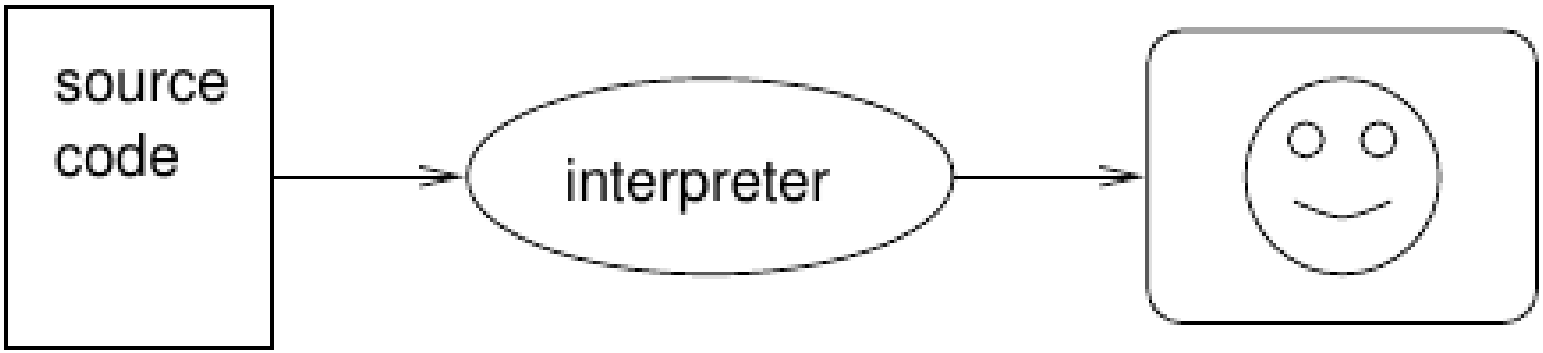
High level languages

- To make programming easier, high level languages can be used.
- A computer can only understand machine language
- Humans wish to write in high level languages
- High level languages have to be re-written (translated) into machine language
- Done by special programs called compilers, interpreters, or assemblers

Outline

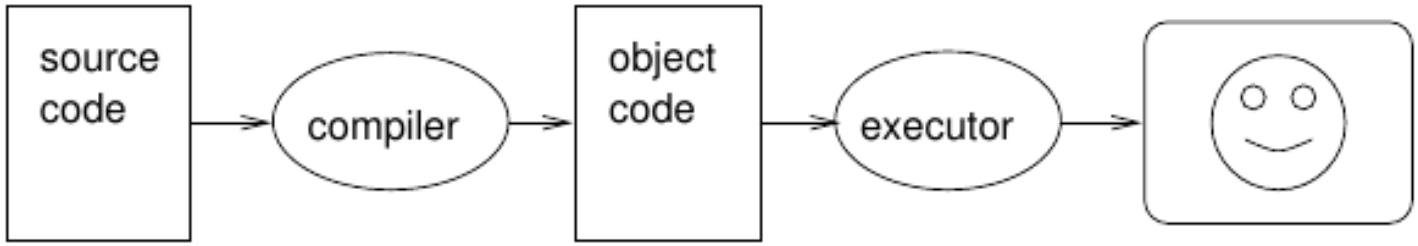
Objectives
Introduction
Hardware
Software
Low level
High level
Interpreted and compiled
C++
Compilers
Editors
Debuggers
Profilers
Overview
Computational thinking
Algorithms
Pseudocode
Basic elements

Interpreted versus compiled languages



The interpreter reads the source code...

... and the result appears on the screen.



The compiler reads the source code...

... and generates object code.

You execute the program (one way or another)...

... and the result appears on the screen.

Outline

Objectives
Introduction
Hardware
Software
Low level
High level
Interpreted and compiled
C++
Compilers
Editors
Debuggers
Profilers
Overview
Computational thinking
Algorithms
Pseudocode
Basic elements

C++

- C++ is a human readable high level language
- The higher-level language code is called source code.
- C++ is designed to be a compiled language
- Generated program is often highly efficient.
- Machine translated code is also called object code or an executable.
- A compiler translates your C++ code into machine readable low level language
- The set of tools is sometimes known as the development tool-chain, whose core are a compiler and its linker.

Outline

1 Objectives

2 Introduction

3 Hardware

4 Software
- Low level
- High level
- Interpreted and compiled
- C++
- Compilers
- Editors
- Debuggers
- Profilers
- Overview

5 Computational thinking
- Algorithms
- Pseudocode
  - Basic elements

Simplified steps of a compiler

1. Checking the syntax:
   - Compiler will check that you are using the language (C++) correctly
   - If you make a mistake, the compiler will issue an error message.
2. Produce the executable file:
   - Produces a file containing the machine code that the computer can execute.
   - You "run" the executable.

The target code may be the machine language for a particular platform or embedded device.

Parts of the development tool-chain

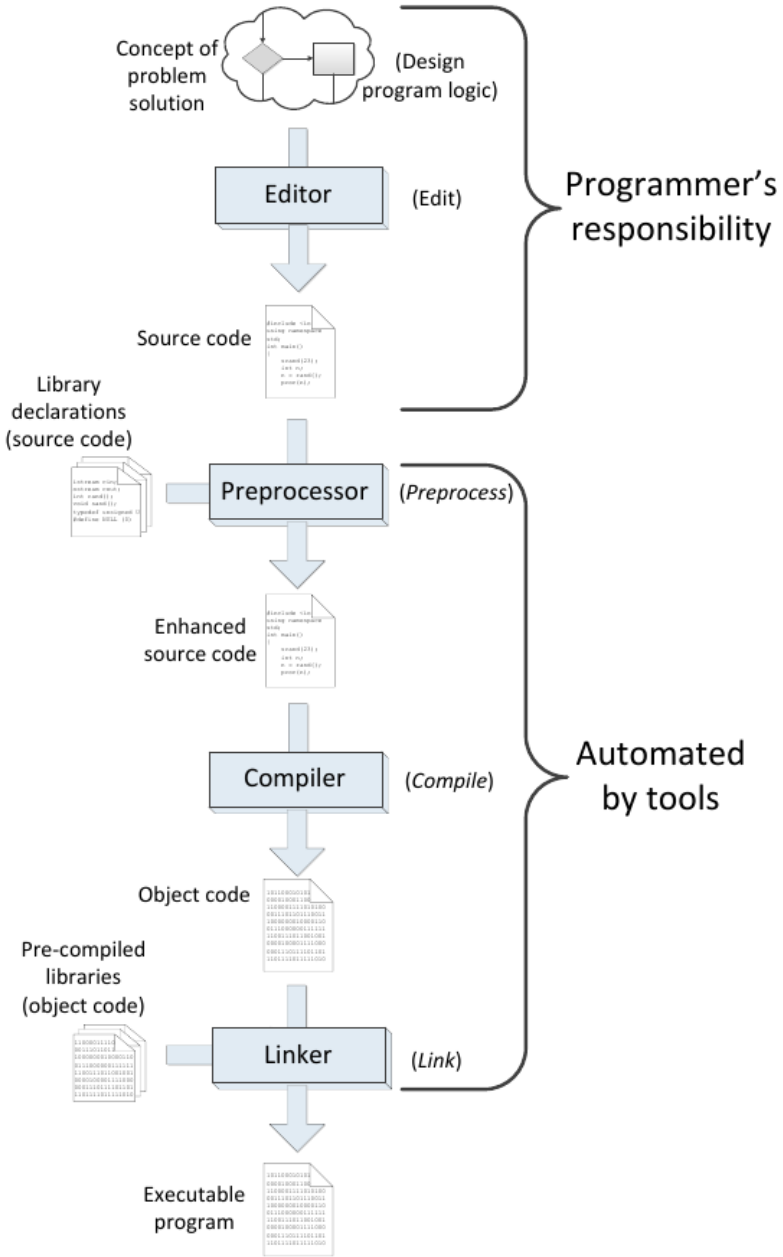
Don't usually think about the C++ pre-processor, compiler, and linker working as three separate programs (although they are):

- **Pre-processor** adds to or modifies the contents of the source file before the compiler begins processing the code. *#include* information about libraries
- **Compiler** translates $C++$ source code to machine code.
- Most compiled $C++$ code is incapable of running by itself and needs some additional machine code to make a complete executable program. The missing machine code has been pre-compiled and stored in a repository of code called a library. **Linker** combines the compiler-generated machine code with pre-compiled library code or compiled code from other sources to make a complete executable program.

Outline

1 Objectives
2 Introduction
3 Hardware
4 Software
Low level
High level
Interpreted and compiled
C++
Compilers
Editors
Debuggers
Profilers
Overview
5 Computational thinking
Algorithms
Pseudocode
Basic elements

Editors and compilers



Editors

- Allows the user to enter the program source code and save it to files.
- Increase programmer productivity by using colors to highlight language features.
- Syntax of a language refers to the way pieces of the language are arranged to make well-formed sentences.
  - "The tall boy runs quickly to the door." uses proper English syntax.
  - "Boy the tall runs door to quickly the." is not correct syntactically, though has correct words.
- Syntax-aware editors can use colors or other special annotations to alert programmers of syntax errors before the program is compiled.

Outline

Objectives
Introduction
Hardware
Software
Low level
High level
Interpreted and compiled
C++
Compilers
Editors
Debuggers
Profilers
Overview
Computational thinking
Algorithms
Pseudocode
Basic elements

Debuggers

- A debugger allows a programmer to more easily trace a program's execution in order to locate and correct errors in the program's implementation.
- With a debugger, a developer can simultaneously run a program and see which line in the source code is responsible for the program's current actions.
- The programmer can watch the values of variables and other program elements to see if their values change as expected.
- Debuggers are valuable for locating errors (also called bugs) and repairing programs that contain errors.
- **Learning to debug and troubleshoot on your own may be one of the most important skills of a good programmer.**

Outline

1 Objectives

2 Introduction

3 Hardware

4 Software
- Low level
- High level
- Interpreted and compiled
- C++
- Compilers
- Editors
- Debuggers
- Profilers
- Overview

5 Computational thinking
- Algorithms
- Pseudocode
  - Basic elements

Profilers

- A profiler collects statistics about a program's execution allowing developers to tune appropriate parts of the program to improve its overall performance.
- A profiler indicates how many times a portion of a program is executed during a particular run, and how long that portion takes to execute.
- Profilers also can be used for testing purposes to ensure all the code in a program is actually being used somewhere during testing.
- The main purpose of profiling is to find the parts of a program that can be improved to make the program run faster.

Outline

1 Objectives

2 Introduction

3 Hardware

4 Software
- Low level
- High level
- Interpreted and compiled
- C++
- Compilers
- Editors
- Debuggers
- Profilers
- Overview

5 Computational thinking
- Algorithms
- Pseudocode
  - Basic elements

The whole process

- Someone has a problem
- They write a C++ solution to the problem
- The compiler translates C++ into machine code
- The program you write will demand memory space for variables, communication from and to the keyboard and monitor, and possibly other uses of peripherals.
- The OS will schedule the use of these resources of the computer.
- The CPU, RAM, disk drives, etc follow those instructions

Outline

1 Objectives

2 Introduction

3 Hardware

4 Software
- Low level
- High level
- Interpreted and compiled
- C++
- Compilers
- Editors
- Debuggers
- Profilers
- Overview

5 Computational thinking
- Algorithms
- Pseudocode
  - Basic elements

What is a program?

- Sequence of instructions that specifies how to perform a computation.
- The instructions (or commands, or statements) look different in different programming languages, but there are a few basic functions that appear in just about every language:
  - **input**: Get data from the keyboard, or a file, or some other device.
  - **output**: Display data on the screen or send data to a file or other device. math: Perform basic mathematical operations like addition and multiplication.
  - **testing**: Check for certain conditions and execute the appropriate sequence of statements.
  - **repetition**: Perform some action repeatedly, usually with some variation.

Outline

Algorithms

- The concepts of computer programming are logical and mathematical in nature.
- In theory, computer programs can be developed without the use of a computer.
- Programmers can describe a procedure using abstract symbols that correspond to the features of real-world programming languages but appear in no real-world programming language.

Algorithms and computational problem solving

Our goals this week include:

- learn how to break apart a big problem into several smaller problems
- Create a sequence of steps for automatically solving these problems
- Pattern these small solutions together in such a way that together, they automatically solve the large problem

Algorithms

What is an algorithm?

- Algorithm is a concise, correct step-by-step sequence of steps to solve a problem.

And why do we care to discuss algorithms here?

- To be successful in writing programs, you need to write, or at least think, algorithms.

Outline

1 Objectives

2 Introduction

3 Hardware

4 Software
- Low level
- High level
- Interpreted and compiled
- C++
- Compilers
- Editors
- Debuggers
- Profilers
- Overview

5 Computational thinking
- Algorithms
- Pseudocode
  - Basic elements

Pseudocode

- Step-by-step verbal outline of your code that you can gradually transcribe into programming language.
- Many programmers use it to plan out the function of an algorithm before setting themselves to the more technical task of coding.
- Informal guide, a tool for thinking through program problems
- Shows how a computing algorithm should and could work.
- Intermediate step in programming, in between the initial planning stage and the stage of writing actual, executable code.
- Allows the designer to focus on the logic of the algorithm without being distracted by details of language syntax.

Good pseudocode practices

- Pseudocode needs to be complete.
- It describe the entire logic of the algorithm so that implementation becomes a rote mechanical task of translating line by line into source code.
- In general, the vocabulary used in the pseudocode should be the vocabulary of the problem domain, not of the implementation domain.

What are the basic constructs of pseudocode?

It has been proven that three basic constructs for flow of control are sufficient to implement any "proper" algorithm.

1. SEQUENCE is a linear progression where one task is performed sequentially after another.
2. WHILE is a loop (repetition) with a simple conditional test at its beginning.
3. IF-THEN-ELSE is a decision (selection) in which a choice is made between two alternative courses of action.

Although these constructs are sufficient, it is often useful to include three more constructs:

1. REPEAT-UNTIL is a loop with a simple conditional test at the bottom.
2. CASE is a multiway branch (decision) based on the value of an expression. CASE is a generalization of IF-THEN-ELSE.
3. FOR is a "counting" loop.

Example problems

How can we look up a word in a dictionary?
How can we sort a stack of cards?

Wrapping it up