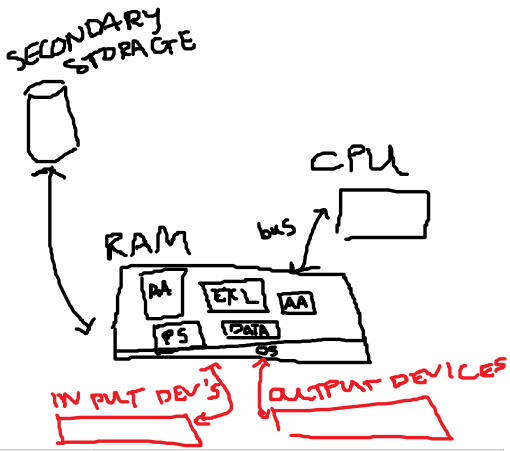
Chapter 1: Introduction to Computers and Programming

Monday, December 18, 2017

Chapter 1: Introduction to Computers and Programming



1.1 Why Program?

- To use computers to perform versatile functions

1.2 Computer Systems: Hardware and Software

- Hardware - physical pieces of the computer

1. CPU	Central Processing Unit, Processor	"Brains"	Fetch (Control Unit), Decode (Control Unit), Execute (Arithmetic Logic Unit)
2. RAM	Random Access Memory, Main Memory, Primary Memory	Volatile storage	Stores portions of currently running programs and data
3. Secondary Storage Devices	Hard drive Mechnical SSD Flash Storage USB drive External hard drive SD card CDs Floppy Disks	Non-volatile storage	Stores information for longer periods of time
4. Input devices	Keyboard, mouse, joystick, touchscreen, stylus, trackpad, webcam, microphone, DJ mixer, sampler, scanner, light sensor, heat sensor		Collect data from outside world
5. Output devices	Monitors, headsets, speakers, vibrating controller, printers		Send data to outside world

1.3 Programs and Programming Languages

Program - set of instructions that the computer follows to perform a task; software

Algorithm - recipe for solving a problem

Set of well-defined steps that terminates

Source Code - contains instructions written in a programming language

Set of well-defined steps that terminates

Source Code - contains instructions written in a programming language

C++ source code
.cpp

compile

Machine Language - Binary language consisting of 0s and 1s (e.g. an executable file)

10111010000000101

Compile - From Source Code to Machine Language

1.4 What is a program made of?

Programming Language contains 5 elements:

1. Keywords	Words that have a special meaning. a.k.a reserved words	namespace using int
2. Programmer-defined Identifiers	Words or names defined by the programmer. Symbolic names that refer to variables or programming routines	hours rate Pay grossPay TAX_RATE
3. Punctuation	Mark the beginning or ending of a statement Or separate items in a list	; { } ( )
4. Operators	Perform operations on (modify) pieces of data	+ - * /
5. Syntax	Rules for combining (1) - (4)	

Visual Studio - This is an IDE (Integrated Development Environment).

C:\Users\Student\source\repos\GrossPay\GrossPay\Payroll.cpp 1

```
1  // This program calculates the user's pay
2  #include <iostream>
3  using namespace std;
4
5  int main() {
6
7     double hours, rate, pay;
8
9     cout << "How many hours did you work?" << endl;
10    cin >> hours;
11
12    cout << "How much do you get paid per hour?" << endl;
13    cin >> rate;
14
15    pay = hours * rate;
16
17    cout << "Pay is " << pay << endl;
18
19    system("pause");
20
21    return 0;
22 }
```

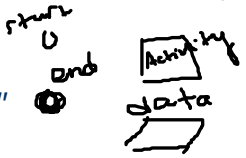
- **1.5 Input, Processing, and Output**

• Input: Data sent to the program	hours, rate
• Processing: Modification of data	pay = hours * rate

• Output: Data sent from the program	pay

1.6 The Programming Process

1. Clearly define what the program is to do.
2. Visualize the program running on the computer.
3. Use design tools (pseudocode, flowchart) to create a model of the program.
   a. *Display "How many hours did you work?".*
   b. *Input hours.*
   c. *Display "How much do you get paid per hour?"*
   d. *Input rate.*
   e. *Store the value of hours times rate in the pay variable.*
   f. *Display the value in the pay variable.*



4. Check the model for logical errors (error that causes the program to produce incorrect results)
5. Type the code, save it, and compile it.
6. Correct any errors found during compilation (compilation errors, compile-time errors, syntax errors)
7. Run the program with test data for input.
8. Correct any errors found while running the program.
   Repeat steps 5 through 8 as many times as necessary.
9. Validate the results of the program

Errors

Syntax / Compile-Time / Compilation Error

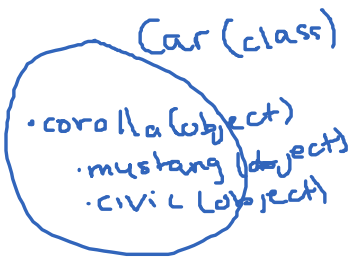
- **Error that causes the program to fail to compile**

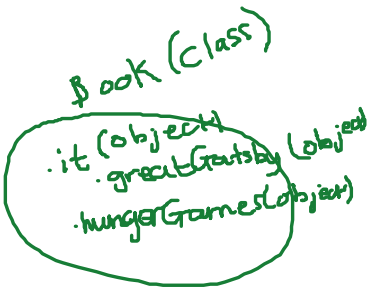
Run-time error

- **Error that occurs after the program is started**
- **Could be a logical error**
- **Could be another type of error**

1.7 Procedural and Object-Oriented Programming

Classes - Categories
Objects - Members of a class





Objects

- a.k.a instances of a class
- Instantiation - Creating an object based on a class

- An object is an instantiation of a class
- Objects have:
  - Data (things they know) - attributes
  - Methods (things they do) - functions, behaviors