CISC 105
Summer 2005

Instructor: Sara Sprenkle
sprenkle@cis.udel.edu
TA: Gang Situ
situ@cis.udel.edu
June 6, 2005

What to Expect from this Class

- First programming course
- Lots to learn!
  - Problem solving
  - Programming environment (UNIX)
  - Programming language (C)
- A Cowboy's Wisdom:
  - Good judgement comes from experience
  - Experiences comes from bad judgement

Class Details

- Monday lectures
  - Quiz at beginning of each class
  - See web page for example code, lecture slides
- Thursday labs (basement of Smith Hall)
  - **Mandatory** attendance
  - Lab due following week (8 labs)
- Participation

Class Details

- 2 Exams
  - Midterm: June 28
  - Final: August 12
- 2 Projects
  - Demos with Gang and me
- Course Project Manager
  - https://atlas.cis.udel.edu:8443/scheduler/group.html

Survey Says…

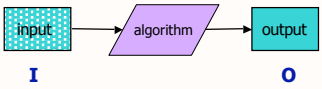
- More about you!

Problem Solving 101

- Computational Problem: Some problem that can be solved by logic
- Algorithm: A well-defined recipe for solving a problem that
  - Has a finite number of steps
  - Completes in a finite amount of time
- Program: An algorithm written in a computer language (Also called code)

More on Algorithms

- Algorithms often have a defined type of input and output.
- Correct algorithms give the intended output for a set of input.
- Example: Multiply by 10
- I/O for a correct algorithm:
  - 5,50
  - .32, 3.2
  - x, 10x



I O

Making a Peanut Butter & Jelly Sandwich

- How do you make a peanut butter and jelly sandwich?
- Write down the steps so that someone else can follow your instructions
  - Make no assumptions about the person's knowledge of PB&J sandwiches

Discussion of PB&J

- Be unambiguous, descriptive
  - Must be clear for the computer to understand
  - "Do what I **meant**! Not what I said!"
- Naming
  - Identify things we're using
- Reusing similar techniques
  - Do the same thing with a little twist
- Looping
  - For repeating the same action

Programming Languages

- Programming language:
  - Specific rules for what is and isn't allowed
  - Must be exact
  - Computer carries out commands as they are given
- Syntax: the symbols given
- Semantics: what it means
- Example: III * IV = 3 x 4 = 12
- Programming languages are unambiguous

C

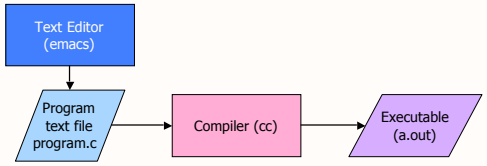
- A common programming language
- Flexible, fast, useful language
- Used by scientists, engineers, systems programmers

Example C Program

```
#include<stdio.h>
int main() {
  int answer;
  printf("Hello, class.\n");
  answer = 2 + 2;
  printf("2+2=%d\n", answer);
  return 0;
}
```

The Programming Process

1. Programmer types a program into a **text editor** (Emacs).
2. A **compiler** (a program itself) turns the program into binary code.
3. **Executable** executes the commands.



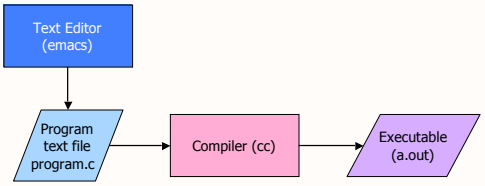
UNIX operating system

- Operating system
  - Manages the computer's resources, e.g., CPU, memory, disk space
  - Examples: UNIX, Windows XP, Windows 2000, Mac X, Linux, etc.
- UNIX
  - Command-line interface (not a GUI)
  - Type commands into terminal window
  - Example commands:
    - cp file1.c file1copy.c (copy a file)
    - mkdir cisc105 (make a directory)

Lab00: Let's Try It!

```
/*
 * Sara Sprenkle 06/06/05
 * first.c
 * In-class example of a simple C program
 */
#include<stdio.h>
int main() {
   printf("Hello, class.\n");
   return 0;
}
```


The Programming Process



In a terminal:
- emacs &
- cc program.c
- ./a.out

The Program

```
/*
 * Sara Sprenkle 06/06/05
 * first.c
 * In-class example of a simple C program
 */
#include<stdio.h>
int main() {
   printf("Hello, class.\n");
   return 0;
}
```

Required in every C program

The Program

```
/*
 * Sara Sprenkle 06/06/05
 * first.c
 * In-class example of a simple C program
 */
#include<stdio.h>
int main() {
   printf("Hello, class.\n");
   return 0;
}
```

} Called the *body* of main

The Program

```
/*
 * Sara Sprenkle 06/06/05
 * first.c
 * In-class example of a simple C program
 */
#include<stdio.h>
int main() {
   printf("Hello, class.\n");
   return 0;
}
```

Comments: describe the program in English

The Program

```
/*
 * Sara Sprenkle 06/06/05
 * first.c
 * In-class example of a simple C program
 */
#include<stdio.h>
int main() {
   printf("Hello, class.\n");
   return 0;
}
```

a statement

0 or more statements make up the body of main

The Program

```
/*
 * Sara Sprenkle 06/06/05
 * first.c
 * In-class example of a simple C program
 */
#include<stdio.h>
int main() {
   printf("Hello, class.\n");
   return 0;
}
```

header file - contains the definition of the function printf

Introduction to Variables

- Variables have names, called **identifiers**
- A variable name (identifier) can be any one word that:
  - Consists of letters, numbers, or _
  - Cannot start with a number
  - Cannot be a C **keyword** (like int or main)
- Remember that C is case-sensitive:
  - change isn't the same as Change

Types

- Variable types are the kind of thing the variable can hold
- Types:
  - **int**
    - integers, e.g., -214, -2, 0, 2, 100, etc.
  - **float**
    - decimal numbers, e.g., .001, 1.234, 1000.1, 0.00
  - **double**
    - more accurate decimal numbers (more places)
  - **char**
    - letters ('a', 'z', 'K'), numbers ('0', '5', '9'), '*', '&', etc

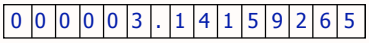
What is the value's type?

Value	Type
'q'	
3.14	
-15.6432	
12	
'?'	
0	
'0'	

How big (or small or precise) can we get?

- We cannot represent all values
- Problem: Computer has a finite capacity
  - The computer only has so much memory that it can devote to one value.
  - Eventually, reach a cutoff
    - Limits size of value
    - Limits precision of value

PI has more decimals, but we're out of space!



In reality, computers represent data in binary, using only 0s and 1s

Declarations

- A declaration is a C statement that sets up a variable.
  - Declares the **type** and **identifier** for the variable
- Like most statements, it must end in a **;**

```
int x;
double my_num;
char option;
```

- You can only declare a variable *once*!

Assignments

- Variables can be given any value that matches their type.
- The C statement that gives a variable its value is called an **assignment** statement.
- After a variable is given a value, the variable is said to be **initialized**.
- These aren't equations! Read "=" as "gets"

```
x = 4;
my_num = 3.4;
option = 'q';
```


Literals

- Pieces of data that are not variables are called **literals**.
- Ex:

```
4
3.2
'q'
```


Arithmetic

- You can use the assignment operator (=) and math operators (*,/,+,-) to do arithmetic.
- Remember your order of operations! (PEMDAS)
- The thing on the left gets changed.

```
x = 4+3*10;
y = 3.0/2.0;
z = x+y;
```

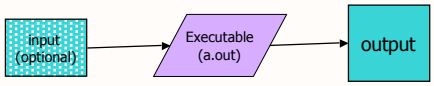
The right-hand sides are **expressions**, just like in math.

Quick Steps

- You can combine a variable declaration and assignment:

```
int x = 0;
double my_num = 3.4;
int num2 = (34/56)*3/2-6;
```

Printing Output



- printf("Hello, class\n");

string literal

- printf("Your answer is %d.\n", answer);

Printing Output

- Format specifiers
  - int -> %d
  - double -> %lf
  - float -> %f
  - char -> %c
- Examples
  - double pi = 3.14159;
  - printf("The value of pi is %lf.\n", pi);
  - printf("The circumference of a circle of radius %d is %lf.\n", num, num*num*pi);

Printf

- Function in stdio.h
- For every format specifier in the string literal, you must have one parameter after the quotes
- printf("%c %d", c, count);
- printf("%c,%d", c, d);
- printf("%lf\n", result);
- printf("%d * %d = %d\n", …);

Examples

- Example 1

```
int a;
char b;
printf(?, a, b);
```

- Example 2

```
int numerator, denominator;
double answer;
printf(?, numerator, denominator, answer);
```


Printing Output

- Escape Sequences
  - newline character (carriage return) -> \n
  - tab -> \t
  - quote -> \"
  - forward slash -> \\
- Examples:
  - printf("%d * %d \t = \t %d\n", 3, 4, 3*4);
  - printf("To print a \\, you must use \"\\\\\"\n");

Examples

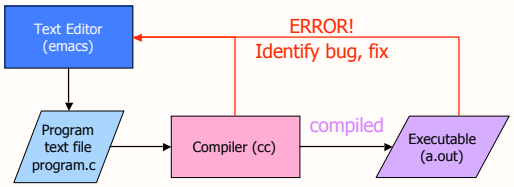
- Print To print a tab, you must use '\t'.
- Print I said, "How are you?"
- Given char carriage_return = '\n';
  - Print the character

ERROR!!

- Sometimes the program doesn't work
- Types of programming errors:
  - Compiler error
    - No semicolon
    - Int v;
  - Logic error
    - answer = 2+3;
    - "The anser is "
  - Runtime error
    - answer = 2/0;

And then what?

- Fix the program, recompile, re-execute until everything works
- The error is often called a "bug"
- Fixing it is called **debugging**



Variables: The Rules

1. You must declare and initialize a variable before using it on the righthand side (rhs) of a rule.
2. Only the variable to the left of the = changes
3. There can be one and only one variable on the lefthand side (lhs).
4. You cannot put data into a variable that does not match the variable's type.
5. You can only have one variable with any given name in a particular block.
6. You must declare all variables before any other statements.

This is NOT Math Class

- Assignment statements are NOT math equations!

  count = count + 1;
- These are commands!

  x = 2;

  y = x;

  x = x + 3;
  - What's the value of y?

Two Types of Division

- Double Division
  - 3.0 / 6.0
  - 6.0 / 3.0
  - x/1.5
  - At least one number must have a decimal
- Integer Division
  - 3/6
  - 6/3
  - x/y, if both x and y are ints
  - Both numbers are integers

Division Practice (NOT Math class)

- int x = 6/4;
- int y = 4/6;
- double z = 4/6;
- double a = 6/4;
- double b = 6/12.0;
- int c = 6.0/12;

Modulo Operator

- Modular Arithmetic: Remainder from division
- Works with integers only
- Operator is % (NOT PERCENT!)
- 6 % 4 is read "six mod four"
- 3 % 6 =
- 7 % 2 =
- 7 % 14 =
- 14 % 7 =

Brainstorm

- What useful thing does % 10 do?
  - 3 % 10 =
  - 51 % 10 =
  - 40 % 10 =
  - 678 % 10 =
- What useful thing does /10 do (integer division?)
  - 3/10 =
  - 51/10 =
  - 40/10=
  - 678/10 =
- What useful thing does % 2 do?

Trick #1: Casting

- To change a variable's type, you can cast from one type to another

```
int x = 4;
double y = 10/(double)x;
```

- int x = 5.5;
- int y = (int)5.5;

Trick #2: Arithmetic Shorthands

- Increment Operator
  - x =x + 1;
  - x++;
- Decrement Operator
  - x = x – 1;
  - x--;
- And others:
  - x += 2;
  - amount *= 1.05;

Practice

- Average three numbers
- Celsius to Fahrenheit
  - F = (9/5)C + 32