Announcements!

- Two more tutorial sessions
  - ➤6:00 Tonight
  - ➤7:00 Tonight
  - ➤Both in Colaboratory 1 in Odegaard Library
- Section Swaps—Get a form if you want to try to switch quiz sections

Announcements!

- Are you on the mailing lists???
- First Quiz Section tomorrow!
- Remember—Class goes until 1:00, not 12:50!

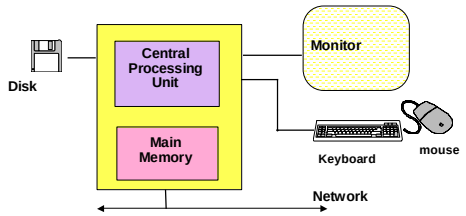
CSE 142
Computer Programming I

Variables, Values, and Types

Chapter 2 Overview

- Chapter 2: Read Sections 2.1–2.6, 2.8
  - ➤Lots of little snippets on various topics
  - ➤We'll fill things out as we go along
- Specifically:
  - ➤Types, variables, values
  - ➤Expressions, assignment
  - ➤Input / Output
  - ➤Programming Style

Review: What's A Computer?



Inside the CPU and Memory

- We've talked about what the CPU does
  - ➤Executes instructions one at a time
  - ➤A series of instructions is a program
- The memory holds the instructions and data for the CPU
  - ➤Organized as a set of numbered locations
  - ➤Each holds one unit of information

Everything is Binary!

- All of the information in the computer is stored as 1s and 0s
  - ➤Integers
  - ➤Real numbers (sort of…)
  - ➤Characters
  - ➤Strings
  - ➤Pictures
  - ➤Programs

How is Data Stored?

- Integers
  1067: 00000000000000000000010000101011
- "Floating Point" numbers
  6.2: 01000000110001100110011001100110
- Characters
  'a': 01100001
- Programs are also coded as numbers

Memory

Address	Contents
0:	01101110
1:	00000000
2:	00000001
3:	10001000
4:	11111111
5:	01110111
6:	00010110

A Program (CPU Instructions)

1. *Set location 4 to 00000001*
2. *Set location 5 to 00000010*
3. *Add the contents of locations 4 and 5 and put the result in location 2*
4. *Print the contents of location 2 as an integer*

Variables

- If programmers had to do everything in binary… *they would go crazy!*
- If they had to remember the memory locations of data… *they would go crazy!*
- Fortunately, we have a tool to help us
  - ➤Variables are names for places in memory
  - ➤Types keep track of the kind of data
- *Programmers still go crazy…*

How to Say it in C

```
#include <stdio.h>
int main(void) {

    int     firstOperand;
    int     secondOperand;
    int     thirdOperand;

    firstOperand = 1;
    secondOperand = 2;
    thirdOperand = firstOperand + secondOperand;
    printf("%d", thirdOperand);

    return 0;
}
```

Key

- Stuff you need in any C program
- Memory allocation ("Declarations of variables")
- Directions for CPU ("Executable instructions" or "C statements")

Important Points

- A memory location is reserved by making a declaration
- Good names are critical!
- Variables must be initialized!
- Instructions are executed in the order they appear, unless we do something special

Another Example

```
#include <stdio.h>
int main(void) {

    int     rectangleLength;
    int     rectangleWidth;
    int     rectangleArea;

    rectangleLength = 10;
    rectangleWidth = 3;
    rectangleArea = rectangleLength * rectangleWidth ;
    printf("%d", rectangleArea);

    return 0;
}
```


"Hand Simulation"

That Was Weird...

- The '=' sign is rather odd
  - In math, the = means two things are equal
  - The developers of C were cruel, wicked fiends, who just wanted to confuse poor 142 students.
- In C, the '=' sign means "gets"
  - It is the assignment operator
  - x = y means "x gets the value of y"

Variable Names

- "Identifiers" are names of things in a program
- In C, there are rules about identifiers:
  - They use numbers, letters, and '_'
  - Cannot be "reserved words"
  - Cannot *begin* with a number
  - They are "case sensitive"
  - Can be as long as you want

Reserved Words

- Certain words are "reserved"
  - They have a special meaning in C
  - Cannot be used for anything else!
  - Must be spelled correctly
  - Also called "keywords"
  - We have already seen **`int`**
  - There are a few dozen in C

Variables and Memory

- Each variable in C names a location in memory
- Each memory location contains 0s and 1s
- What do those 0s and 1s mean?
  - Types tell us

Memory and Types

- Types tell us how to interpret the 0s and 1s in the memory
- Example: the integer 1086324736 is
  01000000110000000000000000000000
  If we read it as a floating point number we get 6.0
- Types help the computer and the programmer keep things straight

What Types Are There?

- Basic types:
  - `int` integer numbers
  - `char` *single* characters
  - `double` numbers with fractional parts
- We'll see more types later in the quarter, but you can do quite a bit with these

Declaring Variables

- `int months;`
  - Can hold integer data, like 6, 12, -170000001
- `double pi;`
  - Can hold floating point representations of numbers, like 3.14159, 2.71828
- `char initial;`
  - Can hold *single* characters, like 'i', 'K', '@'

Assignment Statements

- An assignment statement puts a value into a variable
- The assignment may specify a simple value, or an expression
- Remember '=' means "gets"
- The computer always evaluates what's on the right of the '=', and stores it in the *VARIABLE* on the left

`my_age = my_age+1;`

- This is a statement, not an equation. Is there a difference?
- The expression on the right is evaluated first, and then assigned to the variable on the left.
- What is the result?
- This is different than in math!

Important Message!

Always initialize your variables before you read their values!

Why do you think this is important?

Find the Assignments, Declarations, and Initializations

```
int main (void) {
     double income;  /* ??? */
     income = 35500.00;  /* ??? */
     printf("Old income is %f", income);
     income = 39000.00;  /* ??? */
     printf("After raise: %f", income);
}
```


Problem Solving and Design

- Specify the problem
- Analyze the problem
- Design an algorithm to solve the problem
- Implement the algorithm
- Test and verify that it works
  - ➤Test-debug cycle
- Maintain and update the program

Example Problem: Kilometers to Miles

- Problem?
- Algorithm?
- What types for the data?

Example Problem: Kilometers to Miles

- Problem:
  Convert a distance in kilometers to a distance in miles
- Algorithm?
  ```
  miles = kilometers * .62137;
  ```
- What types for the data?
  ```
  double miles, kilometers;
  ```


Example Continued…

```
#include <stdio.h>
int main(void){
  double miles, kilometers;

  miles = kilometers * .62137;

  return 0;
}
```


Example Continued…

```
#include <stdio.h>
int main(void){
  double miles, kilometers;
  printf("Enter a distance in miles: ");
  scanf("%lf", &kilometers);
  miles = kilometers * .62137;
  printf("That's %f miles!\n", miles);
  return 0;
}
```

Running Our Program

```
Enter a distance in miles: 6
That's 3.728220 miles!
```

- Program Trace:

	kilometers	miles
after declaration	?	?
after first `printf`	?	?
after `scanf`	6	?
after assignment	6	3.738220
after second `printf`	6	3.728220

An Assignment Step-By-Step

```
celsius = (fahrenheit-32.0)*5.0/9.0;
```

- Evaluate right-hand side
  - ➤Find current value of `fahrenheit`
  - ➤Subtract 32.0
  - ➤Multiply by 5.0
  - ➤Divide by 9.0
- Assign the value to `celsius`
  - ➤**The old value is lost!**

Notes on Lecture Examples

- Slides will often leave out details
  - ➤`my_age = my_age + 1;`
- This is legal only if:
  - ➤`my_age` has been declared
  - ➤`my_age` has a proper type
  - ➤this occurs in a legal place in the program
  - ➤it occurs in a legal program
- Use common sense and deductive skills!

Does Terminology Matter?

- We've seen a lot of new words today
- You can *write* a complicated program without using these words…
- But you'll have a hard time *talking* about it!
- Learn the terminology as we go, and get in the habit of using it
  - ➤Your TAs will love you!

Compilers, Linkers, etc.