CSE 142
Computer Programming I

Variables

Overview

Concepts this lecture:

- Variables
- Declarations
- Identifiers and Reserved Words
- Types
- Literals
- Assignment statement
- Variable initialization

Variables

Variables **are program elements that can hold values.**

Example: `int totalChangeToReturn;`

A program consists of a set of variables, and a set of instructions that operate on those variables.

Each variable has

- **A *name*, which you make up**
- **A *type* chosen from a set defined by C**

The *Type* of a Variable

The *type* of a variable determines what values it can contain:

Integers

0, 1, 123456, -22

Real Number

0.5, -2.0, 3.14159, 6.02e23

Single Character

'a', '?', 'N', ' ', '0'

Literals

The *Type* of a Variable

Everyday Name for the Type	The C Name
Integer	int
Real number	double
Single character	char

```
int     months;        /* Create a variable of type integer */
double  pi;            /* Create a variable that holds reals */
char    firstInitial;  /* Create a variable that holds a */
                       /* single character
```

Type Errors

'0' is not 0   0 is not 0.0

```
int numberOfOnes;
numberOfOnes = '0';
```

should be an error

How modern compilers think [e.g., Java]:
I won't even compile something with a type error.

How C compilers think:
Well, okay, if you say so.

BUT THE RESULTS ARE NOT WHAT YOU EXPECT.

Assignment Statements

```
int      area;
Int      length;
Int      width;

length  = 16;
width   = 32;
area    = length * width;
```

An assignment statement stores a value into a variable.

The assignment may specify a simple value to be stored, or an expression

Execution of an assignment statement is done in two distinct steps:

- Evaluate the expression on the right hand side
- Store the value of the expression into the variable named on the left hand side

C-7

myAge = myAge+1

This is a “statement”, not an equation. Is there a difference?

The same variable may appear on both sides of an assignment statement

```
myAge = myAge + 1 ;
```

```
balance = balance + deposit ;
```

The old value of the variable is used to compute the value of the expression before the variable is changed.

C-8

Initializing Variables

There is no such thing as a variable that has “no value”!

```
int    myVariable;
```

There is such a thing as a variable that has a value not determined by the program: an uninitialized variable

C-9

Initialization Rule

General rule: variables should be initialized before their value is used.

Failure to initialize...

is a common source of bugs

is a semantic error, not a syntax error

Variables in a C program are not automatically initialized to anything (not even to 0)!

C-10

Declaring vs Initializing

```
int main (void) {
    double income;                /*declaration of income, not an */
                                  /* assignment or initialization  */
    income = 35500.00;            /* initialization of income,     */
                                  /* assignment to income,         */
                                  /* not a declaration.            */
    printf ("Old income is %f", income);

    income = income * 1.05h;      /*assignment to income, not a    */
                                  /* declaration,or initialization */
    printf ("After raise: %f", income);

    return 0;
}
```

C-11

Example Problem: Life as Capitalism

Problem: Given that

- A college graduate makes $55,000 per year
- A non-college graduate makes $35,000 per year
- College costs $7,500 per year to attend
- College takes 4 years to complete
- You go to college at age 18
- You will retire at age 65

Is your lifetime earnings greater by going to college or not?

C-12

Variable Names

- Are examples of *identifiers* (names chosen by the programmer)
- The names you choose should be meaningful to a human reader
  stuff versus lifetimeEarnings
- Once you've picked a name, how you write it is a matter of *style*
  lifetimeEarnings versus LifetimeEarnings versus
  lifetime_earnings versus lifetime_Earnings …
- C imposes some rules that can bite, but the most important are:
  case (upper and lower) matters
  no embedded blanks (e.g., lifetime earnings)
  no weird punctuation (other than '_')
  can't start with a digit (e.g., '0', '1', etc.)

C-13

Reserved words

Identifiers (like variable names) also can't be *reserved words*

int, double, char, void, return, …

VC++ (and many other systems) use color to help you:

Comments are in green
Reserved words are in blue
Everything else (including identifiers) is in black

C-14

Compile / Link / Run

There are really three steps required to run from your C program ("source file"):

1. Compile
   Translate C instructions into hardware instructions.
   Produces .obj files.
2. Link
   Combine your .obj files and those already created for functions you have used (e.g., printf) to create a file of instructions that can be run.
   Produces a .exe file.
3. Run
   Load your .exe file into RAM and set the CPU to executing it.

VC++ will do all steps needed to run when you ask it to run your program.

C-15

Kinds of Errors

1. Compile Time
   These are *syntax errors* – what you wrote isn't C.
   You can't get any further until you fix these.
   Once fixed, all it means is the what your wrote looks like C, not that your program works.
2. Link Time
   Some identifier used in your program that is supposed to be in a .obj file can't be found.
3. Run Time
   These are *bugs*.
   Either your algorithm is wrong or your implementation of it in C is flawed (or both).

C-16

Next Lecture: Expressions

Each lecture builds on the previous ones, so... be sure you're solid with this material before going on!

C-17