Declaring Variables

Variables

- The most commonly used information in your programs will be variables.
- A variable is a storage location for a piece of data that has a value that can vary.
- The data type will determine the range of values allowable, and the program will assign or modify this value as appropriate.
- To put it another way: Declaring variables allocate memory which has a corresponding data type (e.g. integer) and we give that memory location a name.

The Declaration

All variables are declared by stating the type,
followed by the name you are giving it

Examples:

```
int age;

char firstInitial;

float bankBalance;
```

Types of information

- There are five basic data types associated with variables:
  - int
    - integer: a whole number.
  - float
    - floating point value
    - ie a number with a decimal part.
  - double
    - a double-precision floating point value.
  - char
    - a single character.
  - void
    - valueless special purpose type which we will examine closely in later sections.

Important Things to Keep in Mind

- There is no string type in C and that is important to remember.
- Syntax is short
  - int i, j; // declares a couple of integers
  - float weight; // declares a real number
  - char letter; // a single character
- In order to have strings we use an array of characters. (We get to that soon).
- C is case-sensitive.

Why bother with int?

- float and int are handled differently by the computer.
- Integers are stored in true binary form. 123 is basically stored as " 1111011"
- float can't be stored this way, it has two parts a Mantissa and Exponent, and it is much more work for the computer to work with floats.

Use simplest type.

- Additionally it is good programming practice to use the simplest type possible.
  - This avoids confusion on what the variable is used for.
    - e.g. “int age”, is clearer than “float age”.
  - Can expose errors at compile time.
  - Saves memory.
  - Speeds processing.

Naming variables

- Using meaningful variable names is one of the most important things in making your program readable.
- Your variables should do only one thing and be named accordingly. Rename them as your understanding grows.
- Use mixed case ("camelCase"):
  - e.g. firstName, startTime, isFinished

Hungarian notation

- Professional programmers use prefixes to remind them of the data type of the variable. (i for integer, f for float).
- e.g. iAge; fWeight, iCount.
- This can be useful especially when dealing with different data types.
- Not required for this course, but you may find it really helps, even if use for some of your variables.

More about naming

- Your have to practice between the extremes of giving a name like “x” and “fTheFirstUnknownInTheEquation”.
- Short names are often good but avoid ambiguous abbreviations.
- Don’t be lazy, use “name” instead of “nam” and “month” instead of “mon”.
- By convention use i, j, k for simple looping variables.

The Rules

- Every variable name in C must start with a letter or an underscore.
- Variables should not be longer than 31 characters.
- The rest of the name can consist of letters, numbers and underscore characters (only).
- C recognizes upper and lower case characters as being different.
- Finally, you cannot use any of C's keywords like main, while, switch etc as variable names

Initializing Variables

- Unlike other languages, variables are not initialized in C. Meaning, not given an initial value.
- You should always give your variables an initial value.
- At the top of your program you should have an initialization section which sets the value of each variable declared.
- Or, variables can be initialized as follows:
  - int age = 0;   // gives age a known value

Const keyword

- Sometimes we want to have a variable that doesn't change value.
- We can declare them with a const keyword
  - eg.   const float pi = 3.1415;
- This makes the program more readable
  - circumference = 2 * pi * r;

Const (con't)

- In general we avoid using numbers in our code.
- Sometimes we use const for values we don't expect will change often:
  - const int numProvinces = 10;
  - const int numNHLteams = 30;
- Then if when it changes we only have to change program in one place.

Printing in C

printf

printf

- We use the function printf to write characters to the screen.
- In it’s simplest form we just print out a string:
  - `printf(“Hello, World”);`
- However we want to be able to print out variables and do some formatting.

Printing Variables

- printf can take more than one parameter.
- The first parameter is a format string.
- The following parameters are the variables we wish to print.
- We use placeholders in the format string where we wish the value of the variables to appear.
- Example:
  - `printf(“My age is %d”, age);`

Placeholders Data Type

%d	Integer
%ld	long
%f	float
%s	string
%c	character
%g	double

More Examples

- `printf(“%s, %s”, lastName, firstName);`
- `printf(“Weight is %f”, fWeight);`
- `printf(“This show is brought to you by the letter %c and the number %d”, letter, number);`

Formatting characters

- \n new line (most frequent)
- \t horizontal tab
- The backslash works as an escape character. The function knows to treat the next character differently than it normally would.

Printing special characters

- \'   single quote
- \“   double quote
- \\\\   back slash
- Examples:
- printf(“The song title is \”%s\”\n”, title);
- printf(“\\\\n is used for new line\n”);

Specifying Size

- Often we want to restrict the number of spaces printed.
- We do this by adding a number before the letter of the place holder
- e.g. `printf(“%3d”, age);`
- If the variable value is greater than the number of spaces then the field width expands
- `printf(“age %3d”, 1234)`; prints “age 1234”

Specifying decimals

- `printf(“%f”, 1.0)` prints “1.000000”
- so usually we want to specify the number of decimals.
- e.g. `printf(“Price is $%4.2f\n”, price);` will print a well formatted string.
- 4 would be the total field width, but since it expands if necessary we can often use the placeholder %0.2f whenever we just want to restrict to two decimal points.

Justifying your text

- printf("%-30s\n", "- left justify");
- printf("%30s", "right justify");

```
- left justify
                 right justify
```

//30 Characters wide

- or specify width at runtime
- printf(“%*s\n”, width, “hello”);