ESC101: Introduction to Computing

Input and Output

Variable Declaration

- To communicate to compiler the names and types of the variables used by the program
  - Type tells size of the box to store value
  - Variable must be declared before used
  - Optionally, declaration can be combined with definition (initialization)

```
int count;
int min = 5;
```

Declaration without initialization

Declaration with initialization

Data Types in C

- **int**
  - Bounded integers, e.g. 732 or -5
- **float**
  - Real numbers, e.g. 3.14 or 2.0
- **double**
  - Real numbers with more precision
- **char**
  - Single character, e.g. a or C or 6 or $

Notes on Types: char

- Characters are written with ‘ ’ (quotes)
  - ‘a’, ‘A’, ‘6’, ‘$’
- Case sensitive
  - ‘a’ is not same as ‘A’
- Types distinguish similar looking values
  - Integer 6 is not same as character ‘6’
- Special characters:
  - \n (newline), \’ (quote), \” (double quote), \\ (backslash itself), … and many more
  - NOTE: these are SINGLE CHARACTERS, and have to be enclosed in quotes, as ‘\n’

Character representation

- Characters are represented internally using some number that is based on a standard
- One standard commonly used is ASCII standard
- Different characters are internally represented by different internal numeral representation
  - For e.g. ‘A’ may be 65 and ‘a’ may be 97
  - ‘0’ may be 48 and ‘3’ may be 51

A simple program

- **A program that converts Capital to small characters**

```
# include <stdio.h>
int main(){
   char a = 'D';
   char b =_______;
   printf("__ is now __\n",a, b);
   return 0;
}
```

A simple program

- **A program that converts Capital to small characters**

```
# include <stdio.h>
int main(){
   char a = 'D';
   char b = a-'A'+'a';
   printf("__ is now __\n",a, b);
   return 0;
}
```

A simple program

- **A program that converts Capital to small characters**

```
# include <stdio.h>
int main(){
   char a = 'D';
   char b = a-'A'+'a';
   printf("%c is now %c\n",a, b);
   return 0;
}
```

Another simple program

- **A program that uses multiple types**

```
# include <stdio.h>
int main(){
   char letter = '3';
   int number = _________
   printf("letter __ as a number is
__\n", letter, number);
   return 0;
}
```

Another simple program

- **A program that uses multiple types**

```
# include <stdio.h>
int main(){
   char letter = '3';
   int number = letter - '0';
   printf("letter __ as a number is
__\n", letter, number);
   return 0;
}
```

Another simple program

- **A program that uses multiple types**

```
# include <stdio.h>
int main(){
   char letter = '3';
   int number = letter - '0';
   printf("letter %c as a number is
%d\n", letter, number);
   return 0;
}
```

Assignment Statement

- A simple assignment statement

Variable = Expression;

- Computes the value of the expression on the right hand side expression (RHS), and stores it in the "box" of left hand side (LHS) variable
- = is known as the assignment operator.

Assignment Statement

- Examples

```
x = 10;
ch = 'c';
disc_2 = b*b - 4*a*c;
count = count + 1;
```

If count was 5 before the assignment, it will become 6 after the assignment.

- Evaluation of an assignment stmt:
  - Expression on the RHS of the = operator is first evaluated.
  - Value of the expression is assigned to the variable on the LHS.

Input/Output

- Input: receive data from external sources (keyboard, mouse, sensors)
- Output: produce data (results of computations) (to monitor, printer, projector, …)

Input/Output

- printf function is used to display results to the user. (output)
- scanf function is used to read data from the user. (input)
- Both of these are provided as library functions.
  - #include <stdio.h> tells compiler that these (and some other) functions may be used by the programmer.

Output - printf

list of expressions (separated by comma)

string to be displayed, with placeholders

\n is the newline character.

```
printf("%d kms is equal\nto %f miles\n", km, mi);
```

The string contains placeholders (%d and %f). Exactly one for each expression in the list of expressions.

Placeholder and the corresponding expr must have compatible type.

While displaying the string, the placeholders are replaced with the value of the corresponding expression: first placeholder by value of first expression, second placeholder by value of second expression, and so on.

Using format string in printf

```
printf("a= %d, b= %d, hypotenuse squared =%d", a,b, csquare);
```

Format string marker

Argument marker

1. Format string marker marks the first character of the format string.
2. Argument marker marks the first argument after the format string.
3. Print the character marked by format marker and advance format marker one character at a time until a placeholder (%d)is met or the format string finishes.
4. If format string finishes, we TERMINATE.
5. If the format string marker reads %d, then printf takes the value of the variable at the argument marker and prints it as a decimal integer.
6. Advance argument marker; advance format string marker past %d. Go to step 3.

a= 3, b=4, hypotenuse squared = 25

Input - scanf

Similar to printf: string with placeholders, followed by list of variables to read

& is the *addressof* operator. To be covered later.

```
scanf("%d", &km);
```

Note the & before the variable name. DO NOT FORGET IT.

- String in " " contains only the placeholders corresponding to the list of variables after it.
- Best to use one scanf statement at a time to input value into one variable.

Some Placeholders

Placeholder	Type
%d	int
%f	float
%lf	double
%c	char
%%	literal percent sign (%)

If placeholder and expression/variable type do not match, you may get unexpected results.

Formatting Output of a Program (int)

- When displaying an int value, place a number between the % and d which will specify the number of columns to use for displaying the int value (such as %5d).

```
int x = 2345, y=123;
printf("%d\n",x); //Usual

printf("%6d\n",x); //Display using 6 columns

printf("%6d\n",y); //Note: Right aligned

printf("%2d\n",x); //Less columns, same as %d
```

Output

```
2345
  2345
   123
2345
```

Formatting Output of a Program (float)

- Format placeholder id is %n.mf where
  - n is the total field width (both before and after the decimal point), and
  - m is the number of digits to be displayed after the decimal point

```
float pi = 3.141592;
printf("%f\n",pi); //Usual

printf("%6.2f\n", pi); //2 decimal

printf("%.4f\n",pi); //4 decimal
                     // Note rounding off!
```

Output

```
3.141592
  3.14
3.1416
```

Good and Not so good printf's

```
# include <stdio.h>
int main() {
    float x;
    x=5.67123;
    printf("%f", x);
    return 0;
}
```

Compiles ok

Output

5.671230

```
# include <stdio.h>
int main() {
    float x;
    x=5.67123;
    printf("%d", x);
    return 0;
}
```

Compiles ok

-14227741

Printing a float using %d option is undefined. Result is machine dependent and can be unexpected. AVOID!

C often does not give compilation errors even when operations are undefined. But output may be unexpected!

Comments

- Supplementary information in programs to make understanding easier
  - Only for Humans!
  - Ignored by compilers

Comments in C

- Anything written between /* and */ is considered a comment.

```
diameter = 2*radius; /* diameter of a circle */
```

- Comments cannot be nested.

```
/* I am /* a comment */ but I am not */
```

First */ ends the effect of all unmatched start-of-comments (/*).

Comments in C

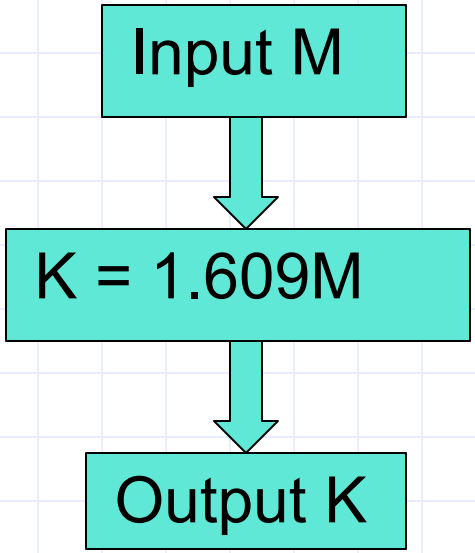
- Anything written after // up to the end of that line

```
diameter = 2*radius; // diameter of a circle
area = pi*radius*radius; // and its area
```

- Not all C compilers support this style of comments.
  - Our lab compiler **does** support it.

An example problem

Problem: Read a distance in miles.
Convert it into kilometres and print it.



Flowchart

Summary: An Example Program

```
#include <stdio.h>
int main()
{
  float mi, km; // decl without initialization

  scanf("%f",&mi); // get miles from user
  km = mi * 1.609; // compute and store km

  printf("%.3f miles = %.3f kms.\n",
          mi, km); // show the answer.
  return 0;
}
```

A simple program

- **A program that converts Capital to small characters**

```
# include <stdio.h>
int main(){
   char a, b;
   scanf("__",__); //input Capital
   char b = a-'A'+'a';
   printf("%c is now %c\n",a, b);
   return 0;
}
```

A simple program

- **A program that converts Capital to small characters**

```
# include <stdio.h>
int main(){
   char a, b;
   scanf("%c",__); //input Capital
   char b = a-'A'+'a';
   printf("%c is now %c\n",a, b);
   return 0;
}
```

A simple program

- **A program that converts Capital to small characters**

```
# include <stdio.h>
int main(){
   char a, b;
   scanf("%c",&a); //input Capital
   char b = a-'A'+'a';
   printf("%c is now %c\n",a, b);
   return 0;
}
```

Another simple program

- **A program that uses multiple types**

```
# include <stdio.h>
int main(){
   char letter;
   int number;
   scanf("__",___);//input
   number = letter - '0';
   printf("letter %c as a number is
%d\n", letter, number);
   return 0;
}
```

Another simple program

- **A program that uses multiple types**

```
# include <stdio.h>
int main(){
   char letter;
   int number;
   scanf("%c",___);//input
   number = letter - '0';
   printf("letter %c as a number is
%d\n", letter, number);
   return 0;
}
```

Another simple program

- **A program that uses multiple types**

```
# include <stdio.h>
int main(){
   char letter;
   int number;
   scanf("%c",&letter);//input
   number = letter - '0';
   printf("letter %c as a number is
%d\n", letter, number);
   return 0;
}
```

ESC101: Introduction to Computing

Operators and Expressions