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 $

More Notes on Types

- Integers (int) are bounded
  - Max value: INT_MAX
  - Min value: INT_MIN
    - These values are system specific
    - -2147483648 … 2147483647 on my machine
- Other data types can only store finite number of values
  - Even some simple real values can not be represented by float and double
- Can surprise you sometimes

```
#include <limits.h>
#include <stdio.h>
int main() {
    printf("Min=%d, Max=%d",
              INT_MIN, INT_MAX);
    return 0;
}
```

```
OUTPUT:  Min=-2147483648, Max=2147483647
```

limits.h contains the definitions of INT_MAX and INT_MIN

1. A statement can span multiple lines.
2. printf can use multiple % placeholders.

```c
#include <stdio.h>
int main() {
    float y = 100000009.0;
    printf("Value of y is %f", y);
    return 0;
}
```

%f is the placeholder for float.

OUTPUT: Value of y is 100000008.000000

Function main

- The point at which C program begins its execution

- Every complete C program must have exactly one main

```
int main () { … }
```

- Returns an int to its caller (Operating System)
  - Return value is generally used to distinguish successful execution (0) from an unsuccessful execution (non 0)

Function main

- Arguments: none ()
  - At least for now

- Body: C statements enclosed inside { and } (to solve the problem at hand)

Tracing the Execution

```
Line
No.
 1   # include <stdio.h>
 2   int main()
 3   {
 4       printf("Welcome to ");
 5       printf("C Programming");
 6       return 0;
     }
```

Output: After lines 3,4   After lines 5,6

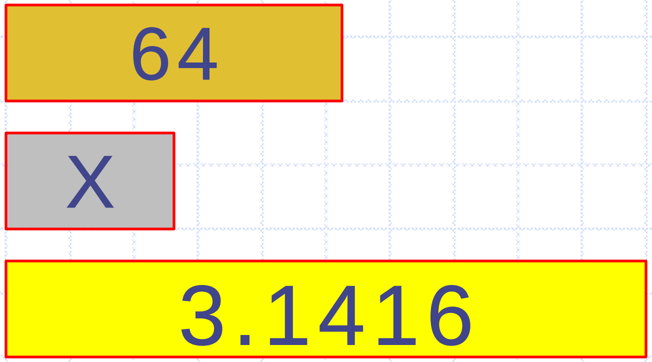
```
Welcome to  C Programming
```

- Program counter starts at the first executable statement of main.
- Line numbers of C program are given for clarity.
- Let us run the program, one step at a time.
- Program terminates gracefully when main ``returns''.

Variables

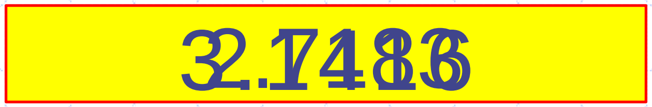
- A name associated with memory cells (box-es) that store data
- Type of variable determines the size of the box.

```
int m = 64;
char c = 'X';
float f = 3.1416;
```



- Variables can change their value during program

```
f = 2.7183;
```



Variable: Box and Value

- Another analogy is that of Envelope and Letter
- Envelope must be big enough to hold the letter!

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

Identifiers

- Names given to different objects
  - Variable, Function etc.
- Consists of letters, digits and underscore (_) symbol
  - Must start with a letter or _

i, count, Lab5, max_Profit, _left, fUnNy

~~5j~~, ~~min Profit~~, ~~lab.7~~

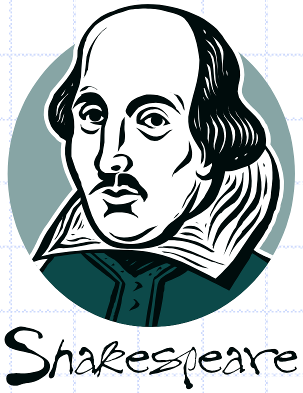
Identifiers

- Certain names are reserved in C
  - Have special meaning
  - Can not be used as identifier
  - Some reserved words: int, float, void, break, switch, const, if, else, ...
- Standard library names should be avoided
  - printf, scanf, strcmp, ...
- Case sensitive
  - Esc101 ≠ esc101 ≠ ESC101

Choosing Identifiers

- Choose meaningful names
  - count vs c vs tmp1
- Should be easy to read and understand
  - count vs c_o_u_n_t
- Shorten only when no loss of meaning
  - Max vs Maximum
- Avoid unnecessary long names
  - a_loop_counter vs counter vs i

“What's in a name? that which we call a rose
By any other name would smell as sweet.”



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.

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 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 %d is met or the format string finishes.
4. If format string finishes, we TERMINATE.

5. If control 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 marker past %d. Go to step 3.

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

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.