ESC101: Introduction to Computing

Types

Reminder: Labs

- Schedule: 2-5 pm
  - B1, B2, B3 : Monday
  - B4, B5, B6 : Tuesday
  - B10, B11, B12 : Wednesday
  - B7,B8,B9: Thursday
- http://iitk.ac.in/doaa/data/coreschedule2016-17-1.pdf
- Location:
  - Core Labs, Room-301 (near DoAA building)
- Labs start on August 1 (Monday).

Tutorials

- Class is divided into 12 sections.
  - B1, B2, …, B12
- Tutorials
  - Tue, 12 noon – 1PM, Tutorial Block.
  - T103(B1)-T112(B10) T203(B11)-T204(B12)

A Simple Program

```
# include <stdio.h>
int main(){
   printf("Welcome to ESC101");
   return 0;
}
```

The program prints the message "Welcome to ESC101"

Another Simple Program

- **Program to add two integers (17 and 23).**

```
# include <stdio.h>
int main(){
    int a = 17;
    int b = 23;
    int c;
    c = a + b;
    printf("Result is %d", c);
    return 0;
}
```

The program prints the message: **Result is 40**

```
# include <stdio.h>
int main ()
{
  int a = 17;
  int b = 23;
  int c;
  c = a + b;

  printf("Result is %d", c);
  return 0;
}
```

This tells the compiler to reserve a "box" large enough to hold an integer value. The box is named "a" for use in the rest of the program.

"= 17" stores value 17 in the box that we have named "a".
It is OK to skip this part and store value later as we do for box named "c".

+ is an operator used to add two numbers. The numbers come from the values stored in the boxes named "a" and "b"

%d tells printf to expect one integer argument whose value is to be printed. We call it placeholder.
We will see more placeholders soon.

Types

- Type:
  - A set of values
  - A set of operations on these values
- We have been using types
  - Natural numbers
    - 1, 2, 3, … values
    - +, - , *, >, <, … operations
  - Complex numbers
    - 5 + 3i, 7 + 2i, …
    - +, - , *, /, conjugate, …
    - **NO** >, < operations

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’

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 sometime

```
#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.

```
#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 100000008.0

Basic Float representation

Machine dependent: One such representation

seeeeeeeemmmmmmmmmmmmmmmmmmmmmmmm

meaning

31 0 bit #

s = sign bit, e = exponent, m = mantissa

number = (sign ? -1:1) * 2^(exponent) * 1.(mantissa bits)

ESC101: Introduction to Computing

Type usage

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 in 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.
- Program terminates gracefully when main ``returns".

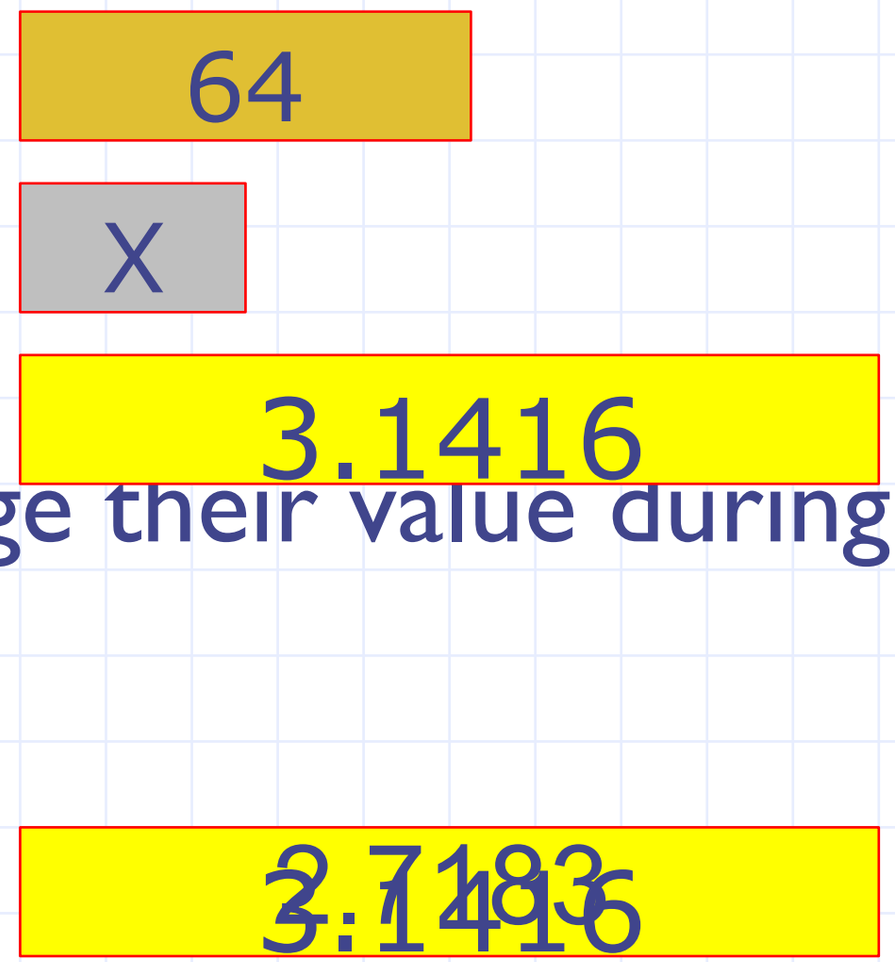
Variables

- A name associated with memory cells (boxes) 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

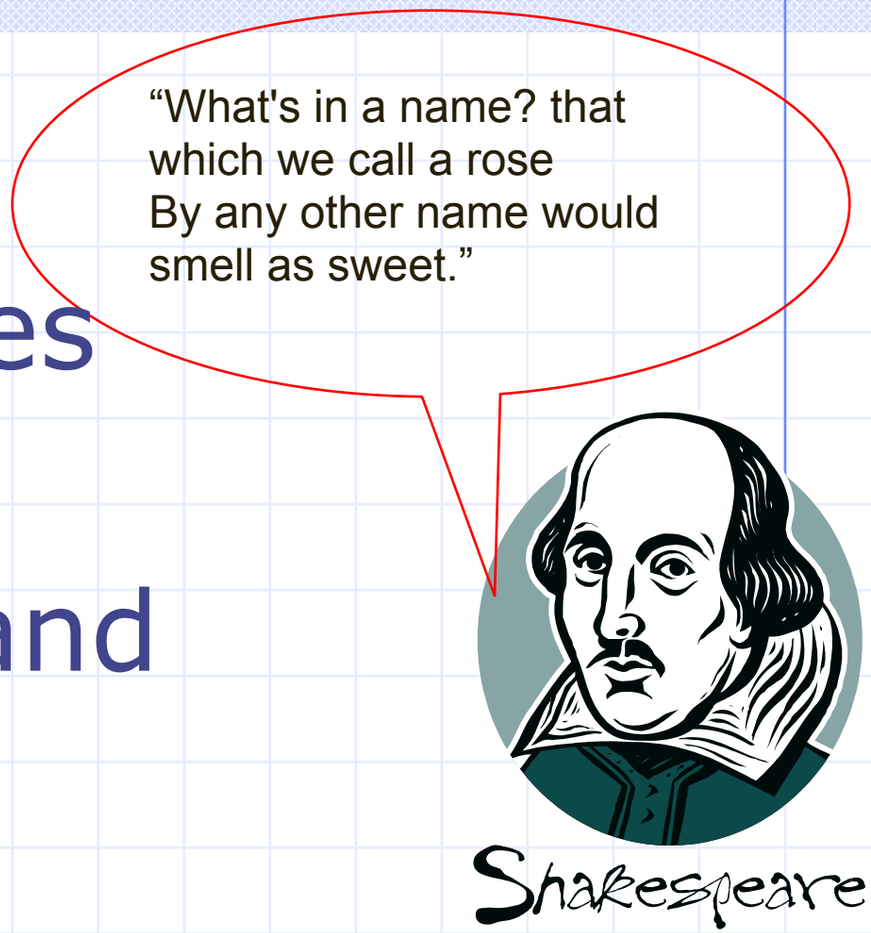
~~5j, min Profit, lab.7~~

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



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;
}
```

Next class

- More on input and output