CS1100 – Introduction to Programming

Lecture 3

Instructor: Shweta Agrawal (shweta.a@cse.iitm.ac.in)

Goals for the day

- Edit, compile and execute the first C program.
- Get simple yet useful tasks done via C programs.
  - Add a set of numbers.
  - Find roots of a quadratic equation.
  - Multiply 2 polynomials.

Goals for the day

- Edit, compile and execute the first C program.
- Get simple yet useful tasks done via C programs.
  - Add a set of numbers.
  - Find roots of a quadratic equation.
  - Multiply 2 polynomials.
- Learn the syntax of C language.
  - Basics – structure of a C program, using standard library.
  - How to store data – variables, data types.
  - How to get inputs, how to print outputs?

Goals for the day

- Edit, compile and execute the first C program.
- Get simple yet useful tasks done via C programs.
  - Add a set of numbers.
  - Find roots of a quadratic equation.
  - Multiply 2 polynomials.
- Learn the syntax of C language.
  - Basics – structure of a C program, using standard library.
  - How to store data – variables, data types.
  - How to get inputs, how to print outputs?
- Learn about the working environment (Linux based OS).
  - editors – gedit and others.
  - compiler – gcc.
  - executing a compiled program.

First C program

```
#include <stdio.h>

/* My first C program */

main() {
    printf("Hello World!\n");
}
```

First C program

```
#include <stdio.h>

/* My first C program */

main() {
    printf("Hello World!\n");
}
```

- stdio.h : standard library of input and output.
- main : a function that every C program must have.
- printf : a useful library function to print several things in C.

First C program

```
#include <stdio.h>

/* My first C program */

main() {
    printf("Hello World!\n");
}
```

- stdio.h : standard library of input and output.
- main : a function that every C program must have.
- printf : a useful library function to print several things in C.

To do anything more useful than merely printing we need to have more operations / commands and storage to store temporary computations.

Variables in C

Add 2 numbers x and y and store the value in z.

- Define two *variables* x and y.
  - What type of values can x and y take?

Variables in C

Add 2 numbers x and y and store the value in z.

- Define two *variables* x and y.
  - What type of values can x and y take?

    integer, positive integers, fractions?

Variables in C

Add 2 numbers x and y and store the value in z.

- Define two *variables* x and y.
  - What type of values can x and y take?

    integer, positive integers, fractions?
  - What do x and y represent?

Variables in C

Add 2 numbers x and y and store the value in z.

- Define two *variables* x and y.
  - What type of values can x and y take?

    integer, positive integers, fractions?
  - What do x and y represent?

    Say marks in Maths and marks in Physics respectively.

Variables in C

Add 2 numbers x and y and store the value in z.

- Define two *variables* x and y.
  - What type of values can x and y take?

    integer, positive integers, fractions?
  - What do x and y represent?

    Say marks in Maths and marks in Physics respectively.
- Use the $+$ operator defined to sum up the values of x and y.

Variables in C

Add 2 numbers x and y and store the value in z.

- Define two *variables* x and y.
  - What type of values can x and y take?

    integer, positive integers, fractions?
  - What do x and y represent?

    Say marks in Maths and marks in Physics respectively.
- Use the $+$ operator defined to sum up the values of x and y.
- Use an assignment operator to store the value in z.

Sum of 2 numbers

```
#include <stdio.h>

/* sum 2 integers */

main() {
    int x;
    int y;
    int z;

    z = x+y;
    printf("%d\n", z);
}
```

Sum of 2 numbers

```
#include <stdio.h>

/* sum 2 integers */

main() {
    int x;
    int y;
    int z;

    z = x+y;
    printf("%d\n", z);
}
```

- int : defines that x, y, z are of type integers.
- z = x+y : evaluates x+y and stores it in z.

Sum of 2 numbers

```
#include <stdio.h>

/* sum 2 integers */

main() {
    int x;
    int y;
    int z;

    z = x+y;
    printf("%d\n", z);
}
```

- int : defines that x, y, z are of type integers.
- z = x+y : evaluates x+y and stores it in z.
- What will be output if we print z?

Sum of 2 numbers

```
#include <stdio.h>

/* sum 2 integers */

main() {
    int x;
    int y;
    int z;

    z = x+y;
    printf("%d\n", z);
}
```

- int : defines that x, y, z are of type integers.
- z = x+y : evaluates x+y and stores it in z.
- What will be output if we print z?
- Initialization or reading of x and y missing.

Sum of 2 numbers

```
#include <stdio.h>

/* sum 2 integers */

main() {
    int x;
    int y;
    int z;

    z = x+y;
    printf("%d\n", z);
}
```

- int : defines that x, y, z are of type integers.
- z = x+y : evaluates x+y and stores it in z.
- What will be output if we print z?
- Initialization or reading of x and y missing.

Sum of 2 numbers – with initialization

```
#include <stdio.h>

/* sum 2 integers */

main() {
    int x = 98;
    int y = 99;
    int z;

    z = x+y;
    printf("%d\n", z);
}
```

- int : defines that x, y, z are of type integers.
- z = x+y : evaluates x+y and stores it in z.
- What will be output if we print z?

Sum of 2 numbers – with initialization

```
#include <stdio.h>

/* sum 2 integers */

main() {
    int x = 98;
    int y = 99;
    int z;

    z = x+y;
    printf("%d\n", z);
}
```

- int : defines that x, y, z are of type integers.
- z = x+y : evaluates x+y and stores it in z.
- What will be output if we print z?

Basic operators in C

- Arithmetic operators: +, -, *, /

Basic operators in C

- Arithmetic operators: +, -, *, /
- Modulus operator: %
  x % y : remainder when x is divided by y.

Basic operators in C

- Arithmetic operators: +, -, *, /
- Modulus operator: %
  x % y : remainder when x is divided by y.
- Assignment operator: =

Basic operators in C

- Arithmetic operators: +, -, *, /
- Modulus operator: %
  x % y : remainder when x is divided by y.
- Assignment operator: =

Input statement: scanf

scanf(format-string, &var1, &var2, ... , &var3);

- scanf is a function which allows us to accept inputs.
- Usually functions take fixed number of parameters/ arguments.
- scanf takes variable number of arguments.
- Notice the **&** preceeding the variables.

Weighted sum of 2 numbers

- Recall x denotes marks in Maths, y denotes marks in Physics.
- We wish to calculate weighted total such that Maths marks are given 30% weightage and Physics marks are given 70% weightage.
- $z = \frac{30}{100}x + \frac{70}{100}y$.

Weighted sum of 2 numbers

```
#include <stdio.h>

/* weighted sum 2 integers */

main() {
    int mathMarks = 98;
    int phyMarks = 99;
    int total;

    total = (30/100)*mathMarks + (70/100)*phyMarks;
    printf("%d\n", total);
}
```

Weighted sum of 2 numbers

```
#include <stdio.h>

/* weighted sum 2 integers */

main() {
    int mathMarks = 98;
    int phyMarks = 99;
    int total;

    total = (30/100)*mathMarks + (70/100)*phyMarks;
    printf("%d\n", total);
}
```

- What is the output of the program?

Weighted sum of 2 numbers

```
#include <stdio.h>

/* weighted sum 2 integers */

main() {
    int mathMarks = 98;
    int phyMarks = 99;
    int total;

    total = (30/100)*mathMarks + (70/100)*phyMarks;
    printf("%d\n", total);
}
```

- What is the output of the program?
- Is the variable total still guaranteed to be an integer?

Weighted sum of 2 numbers

```
#include <stdio.h>

/* weighted sum 2 integers */

main() {
    int mathMarks = 98;
    int phyMarks = 99;
    float total;        /* float variable */

    total = (30/100)*mathMarks + (70/100)*phyMarks;
    printf("%f\n", total); /* change here */
}
```

Weighted sum of 2 numbers

```
#include <stdio.h>

/* weighted sum 2 integers */

main() {
    int mathMarks = 98;
    int phyMarks = 99;
    float total;        /* float variable */

    total = (30/100)*mathMarks + (70/100)*phyMarks;
    printf("%f\n", total); /* change here */
}
```

- What is the output of the program?

Weighted sum of 2 numbers

```
#include <stdio.h>

/* weighted sum 2 integers */

main() {
    int mathMarks = 98;
    int phyMarks = 99;
    float total;        /* float variable */

    total = (30/100)*mathMarks + (70/100)*phyMarks;
    printf("%f\n", total); /* change here */
}
```

- What is the output of the program?
- $\frac{30}{100}$ and $\frac{70}{100}$ evaluate to 0 and therefore total is zero.

Weighted sum of 2 numbers – a correct program

```
#include <stdio.h>

/* weighted sum 2 integers */

main() {
    int mathMarks = 98;
    int phyMarks = 99;
    float total;

    total = (30.0/100)*mathMarks + (70.0/100)*phyMarks;
    printf("%f\n", total);
}
```

Weighted sum of 2 numbers – a correct program

```
#include <stdio.h>

/* weighted sum 2 integers */

main() {
    int mathMarks = 98;
    int phyMarks = 99;
    float total;

    total = (30.0/100)*mathMarks + (70.0/100)*phyMarks;
    printf("%f\n", total);
}
```

- What is the output of the program?

Weighted sum of 2 numbers – a correct program

```
#include <stdio.h>

/* weighted sum 2 integers */

main() {
    int mathMarks = 98;
    int phyMarks = 99;
    float total;

    total = (30.0/100)*mathMarks + (70.0/100)*phyMarks;
    printf("%f\n", total);
}
```

- What is the output of the program?

Learnings so far..

- C allows different kinds of variables to be declared.
- C defines arithmetic operators, like +, -, *, /,...
- Assignment operator "=": used to change contents of a variable.
- Have meaningful names for variables
  mathMarks, phyMarks, total

  choose variable names to be indicative – good programming practice

  avoid reserved words like int, float, .. as variable names.

Exercise: Swap two integers

- Two integers x and y contain 10 and 20 respec.
- Need to exchange values in x and y.
  swap two integers.
- Write a C program to do the same.

Swap – fill in correct code

```
#include<stdio.h>

main() {
    int x, y;

    printf("Enter x:");
    scanf("%d", &x);
    printf("Enter y:");
    scanf("%d", &y);

    /* Fill in code here */

    printf("x = %d\n", x);
    printf("y = %d\n", y);
}
```

Variable modification

- A C program is a sequence of commands that modify different variables using different operators.
- Basic operators in C.
  - Operator precedence and associativity.
- Basic data types in C.
  - How much space does a particular data type take?
  - How to input and output variables of a particular type?

Basic operators in C

- Arithmetic operators: +, -, *, /

Basic operators in C

- Arithmetic operators: +, -, *, /
- Modulus operator: %
  x % y : remainder when x is divided by y.

Basic operators in C

- Arithmetic operators: +, -, *, /
- Modulus operator: %
  x % y : remainder when x is divided by y.
- Assignment operator: =

Basic operators in C

- Arithmetic operators: +, -, *, /
- Modulus operator: %
  x % y : remainder when x is divided by y.
- Assignment operator: =

Operator precedence:

- first: parenthesized sub-expression; inner-most to outer-most.
- second: *, /, % ; left to right.
- third: +, - ; left to right.

Basic operators in C

- Arithmetic operators: +, -, *, /
- Modulus operator: %
  x % y : remainder when x is divided by y.
- Assignment operator: =

Operator precedence:

- first: parenthesized sub-expression; inner-most to outer-most.
- second: *, /, % ; left to right.
- third: +, - ; left to right.
- total = 30 / 100 * mathMarks + 70 / 100 * phyMarks;

Basic operators in C

- Arithmetic operators: +, -, *, /
- Modulus operator: %
  x % y : remainder when x is divided by y.
- Assignment operator: =

Operator precedence:

- first: parenthesized sub-expression; inner-most to outer-most.
- second: *, /, % ; left to right.
- third: +, - ; left to right.
- total = 30 / 100 * mathMarks + 70 / 100 * phyMarks;

  total = ((30 / 100) * mathMarks) + ((70 / 100) * phyMarks);

Basic operators in C

- Arithmetic operators: +, -, *, /
- Modulus operator: %
  x % y : remainder when x is divided by y.
- Assignment operator: =

Operator precedence:

- first: parenthesized sub-expression; inner-most to outer-most.
- second: *, /, % ; left to right.
- third: +, - ; left to right.
- total = 30 / 100 * mathMarks + 70 / 100 * phyMarks;

  total = ((30 / 100) * mathMarks) + ((70 / 100) * phyMarks);
- z = a + b * c * d % e - f / g

Basic operators in C

- Arithmetic operators: +, -, *, /
- Modulus operator: %
  x % y : remainder when x is divided by y.
- Assignment operator: =

Operator precedence:

- first: parenthesized sub-expression; inner-most to outer-most.
- second: *, /, % ; left to right.
- third: +, - ; left to right.
- total = 30 / 100 * mathMarks + 70 / 100 * phyMarks;

  total = ((30 / 100) * mathMarks) + ((70 / 100) * phyMarks);
- z = a + b * c * d % e - f / g

  z = a + (((b * c) * d) % e) - (f / g)

Increment / decrement operators

- ++, - -
- prefix and post-fix only to a variable.

Increment / decrement operators

- ++, - -
- prefix and post-fix only to a variable.

```
#include<stdio.h>

main() {
    int x, y;

    int n = 10;

    x = n++;
    y = ++n;
    printf(" x =  %d, y = %d\n", x, y);

}
```

Increment / decrement operators

- ++, - -
- prefix and post-fix only to a variable.

```
#include<stdio.h>

main() {
    int x, y;

    int n = 10;

    x = n++;
    y = ++n;
    printf(" x =  %d, y = %d\n", x, y);

}
```

Output: x=10, n=12, y=12.