Introduction to Programming

Week – *2*, Lecture – *3*

Introduction to C – Part 1

SAURABH SRIVASTAVA

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

IIT KANPUR

Let’s write some code !!

We have probably “beaten around the bushes” for long

Let’s write some code !!

We have probably “beaten around the bushes” for long

It is the time to see some actual, “executable” code

Let’s write some code !!

We have probably “beaten around the bushes” for long

It is the time to see some actual, “executable” code

Just to reiterate, algorithms and pseudocode are just *preparations* to write code

- Your computer cannot understand that, and hence cannot execute it

Let’s write some code !!

We have probably “beaten around the bushes” for long

It is the time to see some actual, “executable” code

Just to reiterate, algorithms and pseudocode are just *preparations* to write code

- Your computer cannot understand that, and hence cannot execute it

To make your computer do something, you must write a *program* in a language you understand

- … and you must have a software, that can “interpret” it into a language that your computer understands

Let’s write some code !!

We have probably “beaten around the bushes” for long

It is the time to see some actual, “executable” code

Just to reiterate, algorithms and pseudocode are just *preparations* to write code

- Your computer cannot understand that, and hence cannot execute it

To make your computer do something, you must write a *program* in a language you understand

- … and you must have a software, that can “interpret” it into a language that your computer understands

The language here is *C*

- … and that software tool is *gcc*

Shall we solve those quadratic equations?

Procedure QuadraticEquationWithRealRootSolver

```
Inputs: a, b, c

D = b * b - 4 * a * c;
x1 = (-b + √D) / (2 * a)
x2 = (-b - √D) / (2 * a)
return as Output : x1, x2
```

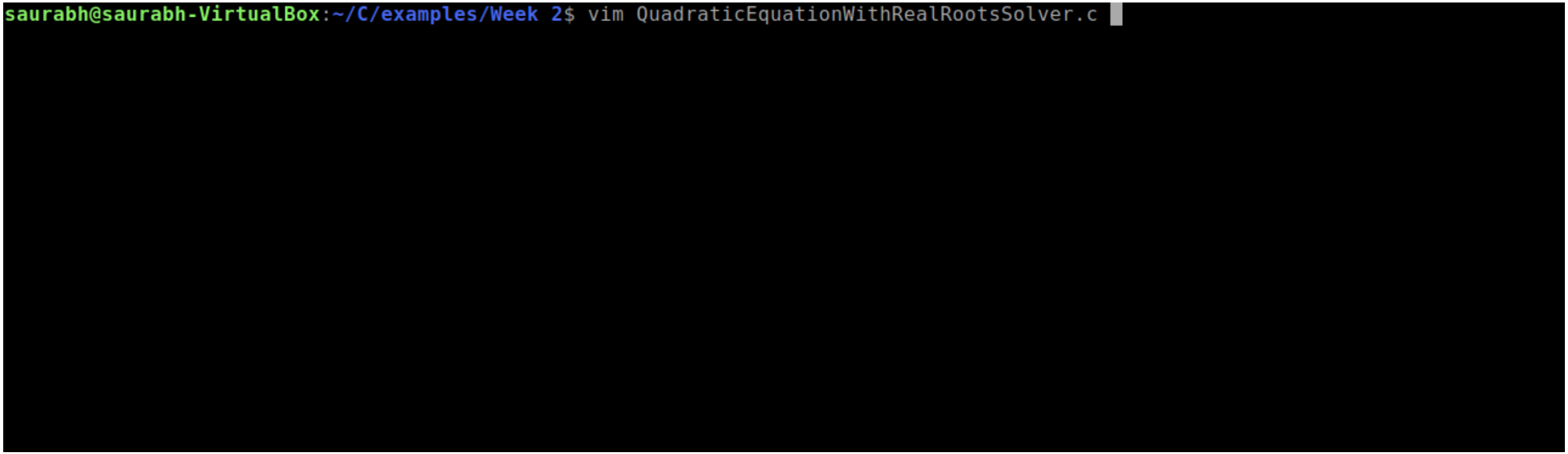
Shall we solve those quadratic equations?

Procedure QuadraticEquationWithRealRootSolver

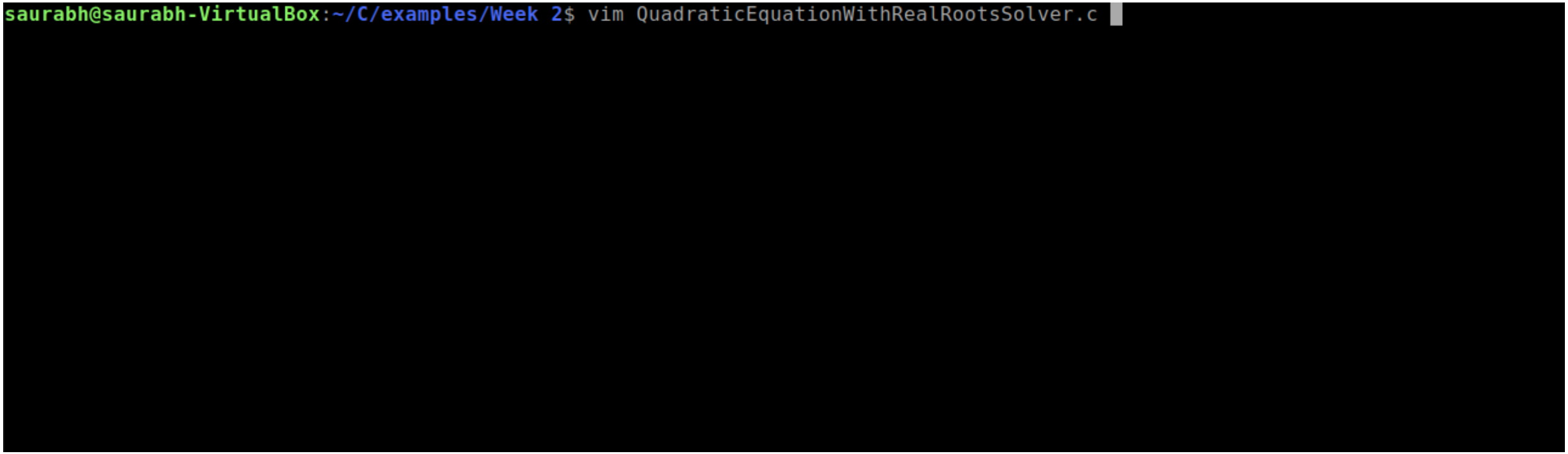
```
Inputs: a, b, c

D = b * b - 4 * a * c;
x1 = (-b + √D) / (2 * a)
x2 = (-b - √D) / (2 * a)
return as Output : x1, x2
```

Assuming that we have to solve only equations with real roots, our pseudocode is very compact

Creating our first C Program !!



Creating our first C Program !!



We begin by creating a new file, with extension `.c`

… and this is our first C Program

```c
#include <stdio.h>
#include <math.h>

int main()
{
        int a = 1;
        int b = 2;
        int c = -15;

        int D = b * b - 4 * a * c;

        double rootD = sqrt(D);

        double x1 = (-b + rootD) / (2 * a);
        double x2 = (-b - rootD) / (2 * a);

        printf("The roots of the equation (%d)x^2 + (%d)x + (%d) = 0 are %lf and %lf", a, b, c, x1, x2);
        return 0;
}
```

… and this is our first C Program

```c
#include <stdio.h>
#include <math.h>

int main()
{
        int a = 1;
        int b = 2;
        int c = -15;

        int D = b * b - 4 * a * c;

        double rootD = sqrt(D);

        double x1 = (-b + rootD) / (2 * a);
        double x2 = (-b - rootD) / (2 * a);

        printf("The roots of the equation (%d)x^2 + (%d)x + (%d) = 0 are %lf and %lf", a, b, c, x1, x2);
        return 0;
}
```

Ignore this part for now !!

… and this is our first C Program

```c
#include <stdio.h>
#include <math.h>

int main()
{
        int a = 1;
        int b = 2;
        int c = -15;

        int D = b * b - 4 * a * c;

        double rootD = sqrt(D);

        double x1 = (-b + rootD) / (2 * a);
        double x2 = (-b - rootD) / (2 * a);

        printf("The roots of the equation (%d)x^2 + (%d)x + (%d) = 0 are %lf and %lf", a, b, c, x1, x2);
        return 0;
}
```

Focus on this part

Inspecting sections of the code

```
6           int a = 1;
7           int b = 2;
8           int c = -15;
```

Inspecting sections of the code

```
6           int a = 1;
7           int b = 2;
8           int c = -15;
```

Let us start here

Inspecting sections of the code

```
6           int a = 1;
7           int b = 2;
8           int c = -15;
```

This is why, a, b and c are variables – they appear here at the LHS of the = sign

Procedure QuadraticEquationWithRealRootSolver

Inputs: a, b, c

```
D = b * b - 4 * a * c;
x1 = (-b + √D) / (2 * a)
x2 = (-b - √D) / (2 * a)
return as Output : x1, x2
```

Inspecting sections of the code

```
6           int a = 1;
7           int b = 2;
8           int c = -15;
```

Look at this statement for example

Inspecting sections of the code

```
6          int a = 1;
7          int b = 2;
8          int c = -15;
```

Look at this statement for example

What this statement essentially instructs your computer to do is:

1. Reserve some space in main memory to store one **integer**
2. Create a name for that location – **a**; from now on, every time I say “a”, just understand I am talking about this memory location.
3. Store the integer **1** at that memory location

Inspecting sections of the code

```
12          double rootD = sqrt(D);
```

… similarly, this statement reserves space in memory to store a real number (one that has a decimal place)

Inspecting sections of the code

```
12          double rootD = sqrt(D);
```

… similarly, this statement reserves space in memory to store a real number (one that has a decimal place)

The value that is stored at that location, is returned by executing a *library function* – `sqrt()`

Inspecting sections of the code

```
12          double rootD = sqrt(D);
```

… similarly, this statement reserves space in memory to store a real number (one that has a decimal place)

The value that is stored at that location, is returned by executing a *library function* – `sqrt()`

Library functions are made available to you “on demand” (not by default)

Inspecting sections of the code

```
2 #include <math.h>
```

… and this is basically the “demand”

Inspecting sections of the code

```
2 #include <math.h>
```

… and this is basically the “demand”

While you write your own programs in files with extension `.c`, library functions are available in files with extension `.h`, also called *header files*

Inspecting sections of the code

```
2 #include <math.h>
```

… and this is basically the “demand”

While you write your own programs in files with extension `.c`, library functions are available in files with extension `.h`, also called *header files*

Usually, one header file contains many library functions

Inspecting sections of the code

```
10          int D = b * b - 4 * a * c;
```

```
14          double x1 = (-b + rootD) / (2 * a);
15          double x2 = (-b - rootD) / (2 * a);
```

… and these are examples of computation

Inspecting sections of the code

```
10          int D = b * b - 4 * a * c;
```

```
14          double x1 = (-b + rootD) / (2 * a);
15          double x2 = (-b - rootD) / (2 * a);
```

… and these are examples of computation

We do not have “different” types of brackets, like small, curly and large – we only have the small brackets, called **parentheses**

Inspecting sections of the code

```
10          int D = b * b - 4 * a * c;
```

```
14          double x1 = (-b + rootD) / (2 * a);
15          double x2 = (-b - rootD) / (2 * a);
```

… and these are examples of computation

We do not have “different” types of brackets, like small, curly and large – we only have the small brackets, called **parentheses**

We will have a look in the next lecture, where you should ideally use them (and where they are not really required)

Inspecting sections of the code

```
printf("The roots of the equation (%d)x^2 + (%d)x + (%d) = 0 are %lf and %lf", a, b, c, x1, x2);
```

… this is an invocation of another library function – one that can show some text on the screen

Inspecting sections of the code

```
printf("The roots of the equation (%d)x^2 + (%d)x + (%d) = 0 are %lf and %lf", a, b, c, x1, x2);
```

… this is an invocation of another library function – one that can show some text on the screen

The method `printf()` can either print some text as it is, or you can use it to print values of the variables that you have in your program

Inspecting sections of the code

```
1 #include <stdio.h>
```

… this is an invocation of another library function – one that can show some text on the screen

The method `printf()` can either print some text as it is, or you can use it to print values of the variables that you have in your program

… and this the header file, which we are “including” in our program to use `printf()`

Inspecting sections of the code

```
1 #include <stdio.h>
```

… this is an invocation of another library function – one that can show some text on the screen

The method `printf()` can either print some text as it is, or you can use it to print values of the variables that you have in your program

… and this the header file, which we are “including” in our program to use `printf()`

Inspecting sections of the code

```
4 int main()
```

This leaves just this statement that we haven’t yet talked about

Inspecting sections of the code

```
4 int main()
```

This leaves just this statement that we haven't yet talked about

Consider `main()` as a special procedure – the procedure which contains the code of your program

Inspecting sections of the code

```
4 int main()
```

This leaves just this statement that we haven't yet talked about

Consider `main()` as a special procedure – the procedure which contains the code of your program

From `main()` you can invoke other procedures (for instance, if we had another procedure for calculating Discriminant separately)

Inspecting sections of the code

```
4 int main()
```

This leaves just this statement that we haven't yet talked about

Consider `main()` as a special procedure – the procedure which contains the code of your program

From `main()` you can invoke other procedures (for instance, if we had another procedure for calculating Discriminant separately)

We will discuss main() at length, later… for now, just understand that all the code you write, lies "inside" the `main()` procedure

The `printf()` Library Function (1/2)

A common term that is used to show text on the screen is “printing text”

The `printf()` Library Function (1/2)

A common term that is used to show text on the screen is “printing text”

If you want to print a “static message”, such as “Hello !”, you can use *pass* this to `printf()`

- i.e. `printf("Hello !");`

The `printf()` Library Function (1/2)

A common term that is used to show text on the screen is “printing text”

If you want to print a “static message”, such as “Hello !”, you can use *pass* this to `printf()`

- i.e. `printf(“Hello !”);`

The term we use for text in the programming world is **String**

- `“Hello !”`, thus, is a string

The `printf()` Library Function (1/2)

A common term that is used to show text on the screen is “printing text”

If you want to print a “static message”, such as “Hello !”, you can use *pass* this to `printf()`

- i.e. `printf(“Hello !”);`

The term we use for text in the programming world is **String**

- `“Hello !”`, thus, is a string

In C (and many other programming languages), strings must be put in quotes, i.e. between a “ and “

The `printf()` Library Function (1/2)

A common term that is used to show text on the screen is “printing text”

If you want to print a “static message”, such as “Hello !”, you can use *pass* this to `printf()`

- i.e. `printf(“Hello !”);`

The term we use for text in the programming world is **String**

- `“Hello !”`, thus, is a string

In C (and many other programming languages), strings must be put in quotes, i.e. between a “ and “

If you want to print the value of a variable (which can be anything, based on your program):

- You create a “placeholder” in your string, where the actual value of the variable will be “pasted” by `printf()`

The `printf()` Library Function (1/2)

A common term that is used to show text on the screen is “printing text”

If you want to print a “static message”, such as “Hello !”, you can use *pass* this to `printf()`

- i.e. `printf(“Hello !”);`

The term we use for text in the programming world is **String**

- `“Hello !”`, thus, is a string

In C (and many other programming languages), strings must be put in quotes, i.e. between a “ and “

If you want to print the value of a variable (which can be anything, based on your program):

- You create a “placeholder” in your string, where the actual value of the variable will be “pasted” by `printf()`

```
printf("The roots of the equation (%d)x^2 + (%d)x + (%d) = 0 are %lf and %lf", a, b, c, x1, x2);
```

The `printf()` Library Function (1/2)

A common term that is used to show text on the screen is “printing text”

If you want to print a “static message”, such as “Hello !”, you can use *pass* this to `printf()`

- i.e. `printf("Hello !");`

The term we use for text in the programming world is **String**

- `"Hello !"`, thus, is a string

In C (and many other programming languages), strings must be put in quotes, i.e. between a “ and “

If you want to print the value of a variable (which can be anything, based on your program):

- You create a “placeholder” in your string, where the actual value of the variable will be “pasted” by `printf()`

```
printf("The roots of the equation (%d)x^2 + (%d)x + (%d) = 0 are %lf and %lf", a, b, c, x1, x2);
```

- Here, `%d` and `%lf` are placeholders, for integers and real numbers respectively

The `printf()` Library Function (1/2)

A common term that is used to show text on the screen is “printing text”

If you want to print a “static message”, such as “Hello !”, you can use *pass* this to `printf()`

- i.e. `printf(“Hello !”);`

The term we use for text in the programming world is **String**

- `“Hello !”`, thus, is a string

In C (and many other programming languages), strings must be put in quotes, i.e. between a “ and “

If you want to print the value of a variable (which can be anything, based on your program):

- You create a “placeholder” in your string, where the actual value of the variable will be “pasted” by `printf()`

```
printf("The roots of the equation (%d)x^2 + (%d)x + (%d) = 0 are %lf and %lf", a, b, c, x1, x2);
```

- Here, `%d` and `%lf` are placeholders, for integers and real numbers respectively
- … and we provide the name of the variables, in that order, after the String

The `printf()` Library Function (2/2)

The `printf()` function takes “at least” one *argument*

The `printf()` Library Function (2/2)

The `printf()` function takes “at least” one *argument*

Arguments are “inputs to a function”

- For example, `sqrt()` function accepts *exactly one argument* – the number whose square root is required

The `printf()` Library Function (2/2)

The `printf()` function takes "at least" one *argument*

Arguments are "inputs to a function"

- For example, `sqrt()` function accepts *exactly one argument* – the number whose square root is required

If a function takes more than one argument, we separate them by commas

The `printf()` Library Function (2/2)

The `printf()` function takes "at least" one *argument*

Arguments are "inputs to a function"

- For example, `sqrt()` function accepts *exactly one argument* – the number whose square root is required

If a function takes more than one argument, we separate them by commas

The **order** of the **arguments matter**

- Thus, if you are invoking a function like `func(x,y)`, it is different than invoking it like `func(y,x)`

The `printf()` Library Function (2/2)

The `printf()` function takes "at least" one *argument*

Arguments are "inputs to a function"

- For example, `sqrt()` function accepts *exactly one argument* – the number whose square root is required

If a function takes more than one argument, we separate them by commas

The **order** of the **arguments matter**

- Thus, if you are invoking a function like `func(x,y)`, it is different than invoking it like `func(y,x)`

With `printf()` the variables that you supply as arguments after the string must follow an order

The `printf()` Library Function (2/2)

The `printf()` function takes "at least" one *argument*

Arguments are "inputs to a function"

- For example, `sqrt()` function accepts *exactly one argument* – the number whose square root is required

If a function takes more than one argument, we separate them by commas

The **order** of the **arguments matter**

- Thus, if you are invoking a function like `func(x,y)`, it is different than invoking it like `func(y,x)`

With `printf()` the variables that you supply as arguments after the string must follow an order

- This order should match the "placeholders" in the string
- These placeholders, also called format specifiers, are fixed for a particular type of value

The `printf()` Library Function (2/2)

The `printf()` function takes "at least" one *argument*

Arguments are "inputs to a function"

- For example, `sqrt()` function accepts *exactly one argument* – the number whose square root is required

If a function takes more than one argument, we separate them by commas

The **order** of the **arguments matter**

- Thus, if you are invoking a function like `func(x,y)`, it is different than invoking it like `func(y,x)`

With `printf()` the variables that you supply as arguments after the string must follow an order

- This order should match the "placeholders" in the string
- These placeholders, also called format specifiers, are fixed for a particular type of value
- For integers, it is `%d`, and for real numbers it is `%lf`
- The variables that we provide as arguments, must honour the order in which format specifiers are provided

“Running” the first C Program (1/3)

```
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ vim QuadraticEquationWithRealRootsSolver.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ gcc QuadraticEquationWithRealRootsSolver.c
```

“Running” the first C Program (1/3)

```
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ vim QuadraticEquationWithRealRootsSolver.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ gcc QuadraticEquationWithRealRootsSolver.c
```

Executing a program is also called “running” the program

“Running” the first C Program (1/3)

```
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ vim QuadraticEquationWithRealRootsSolver.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ gcc QuadraticEquationWithRealRootsSolver.c
```

Executing a program is also called “running” the program

But your program cannot be executed directly by your computer

“Running” the first C Program (1/3)

```
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ vim QuadraticEquationWithRealRootsSolver.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ gcc QuadraticEquationWithRealRootsSolver.c
```

Executing a program is also called “running” the program

But your program cannot be executed directly by your computer

You have to use a software tool called “Compiler” to convert it into a form, that your computer can execute

“Running” the first C Program (1/3)

```
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ vim QuadraticEquationWithRealRootsSolver.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ gcc QuadraticEquationWithRealRootsSolver.c
```

Executing a program is also called “running” the program

But your program cannot be executed directly by your computer

You have to use a software tool called “Compiler” to convert it into a form, that your computer can execute

The compiler for C that we will use, is `gcc`

“Running” the first C Program (1/3)

```
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ vim QuadraticEquationWithRealRootsSolver.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ gcc QuadraticEquationWithRealRootsSolver.c
```

Executing a program is also called “running” the program

But your program cannot be executed directly by your computer

You have to use a software tool called “Compiler” to convert it into a form, that your computer can execute

The compiler for C that we will use, is `gcc`

The syntax to use `gcc` is: `gcc <name of a C code file>`

“Running” the first C Program (2/3)

```
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ vim QuadraticEquationWithRealRootsSolver.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ gcc QuadraticEquationWithRealRootsSolver.c
/tmp/ccqSQGjX.o: In function `main':
QuadraticEquationWithRealRootsSolver.c:(.text+0x3d): undefined reference to `sqrt'
collect2: error: ld returned 1 exit status
```

… well… usually it will work !! But this time, we need a little more

The problem is that `sqrt()` library function !! Actually, `gcc` doesn’t know where to find it…

Basically, it doesn’t know what “library” does it need to bring, while compiling your code… but you can give that information

“Running” the first C Program (2/3)

```
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ vim QuadraticEquationWithRealRootsSolver.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ gcc QuadraticEquationWithRealRootsSolver.c
/tmp/ccqSQGjX.o: In function `main':
QuadraticEquationWithRealRootsSolver.c:(.text+0x3d): undefined reference to `sqrt'
collect2: error: ld returned 1 exit status
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ gcc QuadraticEquationWithRealRootsSolver.c -lm
```

… well… usually it will work !! But this time, we need a little more

The problem is that `sqrt()` library function !! Actually, `gcc` doesn’t know where to find it…

Basically, it doesn’t know what “library” does it need to bring, while compiling your code… but you can give that information

… and this is how you do it !!

“Running” the first C Program (2/3)

```
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ vim QuadraticEquationWithRealRootsSolver.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ gcc QuadraticEquationWithRealRootsSolver.c
/tmp/ccqSQGjX.o: In function `main':
QuadraticEquationWithRealRootsSolver.c:(.text+0x3d): undefined reference to `sqrt'
collect2: error: ld returned 1 exit status
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ gcc QuadraticEquationWithRealRootsSolver.c -lm
```

… well… usually it will work !! But this time, we need a little more

The problem is that `sqrt()` library function !! Actually, `gcc` doesn’t know where to find it…

Basically, it doesn’t know what “library” does it need to bring, while compiling your code… but you can give that information

… and this is how you do it !! Here `-l` is a switch for `gcc`, and `m` is the shorthand for the math library

“Running” the first C Program (3/3)

```
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ vim QuadraticEquationWithRealRootsSolver.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ gcc QuadraticEquationWithRealRootsSolver.c
/tmp/ccqSQGjX.o: In function `main':
QuadraticEquationWithRealRootsSolver.c:(.text+0x3d): undefined reference to `sqrt'
collect2: error: ld returned 1 exit status
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ gcc QuadraticEquationWithRealRootsSolver.c -lm
```

Actually, `gcc` does not “link” all the libraries that come with it by default, when it compiles a program

“Running” the first C Program (3/3)

```
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ vim QuadraticEquationWithRealRootsSolver.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ gcc QuadraticEquationWithRealRootsSolver.c
/tmp/ccqSQGjX.o: In function `main':
QuadraticEquationWithRealRootsSolver.c:(.text+0x3d): undefined reference to `sqrt'
collect2: error: ld returned 1 exit status
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ gcc QuadraticEquationWithRealRootsSolver.c -lm
```

Actually, `gcc` does not “link” all the libraries that come with it by default, when it compiles a program

“Linking” is a process, where you can supply libraries written by other developers, for use with your code

“Running” the first C Program (3/3)

```
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ vim QuadraticEquationWithRealRootsSolver.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ gcc QuadraticEquationWithRealRootsSolver.c
/tmp/ccqSQGjX.o: In function `main':
QuadraticEquationWithRealRootsSolver.c:(.text+0x3d): undefined reference to `sqrt'
collect2: error: ld returned 1 exit status
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ gcc QuadraticEquationWithRealRootsSolver.c -lm
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ ./a.out
```

Actually, `gcc` does not “link” all the libraries that come with it by default, when it compiles a program

“Linking” is a process, where you can supply libraries written by other developers, for use with your code

The “translated code”, which your computer can understand, is saved to a file called `a.out` in the current directory

“Running” the first C Program (3/3)

```
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ vim QuadraticEquationWithRealRootsSolver.c
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ gcc QuadraticEquationWithRealRootsSolver.c
/tmp/ccqSQGjX.o: In function `main':
QuadraticEquationWithRealRootsSolver.c:(.text+0x3d): undefined reference to `sqrt'
collect2: error: ld returned 1 exit status
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ gcc QuadraticEquationWithRealRootsSolver.c -lm
saurabh@saurabh-VirtualBox:~/C/examples/Week 2$ ./a.out
The roots of the equation (1)x^2 + (2)x + (-15) = 0 are 3.000000 and -5.000000saurabh@saurabh-VirtualBox:~/C/examples/Week 2$
```

Actually, `gcc` does not “link” all the libraries that come with it by default, when it compiles a program

“Linking” is a process, where you can supply libraries written by other developers, for use with your code

The “translated code”, which your computer can understand, is saved to a file called `a.out` in the current directory

Which, when called on shell, shows you the “output” of the program !!

Homework !! (They'll get intense now…)

Since the memory allocated to a variable is "limited", there are some restrictions on the values

- For example, a variable of type `int`, can only store `65536` possible integers (can you find out why?)
- Find out more about the range of other types of variables

Also, see if you have any control over "which" 65536 values it would be !!

- Is it possible to "store more positive numbers", if you do not require the variable to "store negative values"
- See if you can get some hints here: https://www.cs.utah.edu/~germain/PPS/Topics/unsigned_integer.html

There are many types of variables that you can create in C, in addition to `int` and `double`

- Read about other data types in C
- Also, find out their associated format specifier to use, with the `printf()` function
- A good starting point is this link: https://www.programiz.com/c-programming/c-data-types

Additional Reading

See if you can grasp something from the discussions at this page https://stackoverflow.com/questions/10409032/why-am-i-getting-undefined-reference-to-sqrt-error-even-though-i-include-math