

ME 172

# Introduction to C Programming Language

**Saif Al-Afsan Shamim**

Lecturer, Dept. of ME, BUET

Courtesy:

Cyrus Ashok Arupratan Atis

M Aminul Hoque

Partha Kumar Das

# ME 172

Please go to the following link and complete registration

<http://bit.ly/2x0jEBX>

# Getting Started

- Create a folder named **ME172** in your **Desktop**
- Inside this folder again create a folder named with your roll no. in the following format  
**1710001**
- Save all your codes in that particular folder in each class
- No one other than yourself will be held accountable if the folder is missing or your codes are not saved inside that folder.
- **The use of Mobile phones/pen drives is strictly prohibited during the class time**

# How C Works

- Executing a program written in C involves following steps:

1. Creating the program (**Editor**)

2. Compiling the program (**Compiler**)

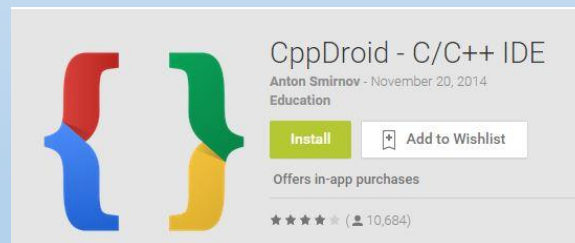
3. Linking the program with functions that are needed from the C library  
(**Linker**)

4. Executing the program

# Compiler(s):

Used for converting Source code into object code(executable program)

- Code Blocks 13.12 for Windows 7/8
- Download  
Link:<http://www.codeblocks.org/downloads/binaries#windows>
- For the peoples who want to run their codes **on the go** try the **CppDroid** app



# Downloading Code::Blocks

The screenshot shows a web browser window with the address bar displaying `www.codeblocks.org/downloads/binaries#windows`. The website header features the Code::Blocks logo and the tagline "Code::Blocks - The IDE with all the features you need, having a consistent look, feel and operation across platforms." A navigation menu includes Home, Features, Downloads, Forums, and Wiki. The main content area is titled "Main" and contains a sidebar with navigation links (Home, Features, Screenshots, Downloads, Binaries, Changelog, Source, SVN, Plugins, User manual, Licensing, Donations) and a "Quick links" section (FAQ, Wiki, Forums, Forums (mobile), Nightlies, Ticket System, Browse SVN, Browse SVN log). The main content area includes a "Please select a setup package depending on your platform:" section with a list of operating systems: Windows 2000/XP/Vista/7/8, Linux 32-bit, Linux 64-bit, and Mac OS X. Below this is a "NOTE" about nightly builds, a "MIRRORS" section, an "IMPORTANT NOTE" about download errors, and another "NOTE" about the changelog for version 13.12. A section titled "Windows 2000 / XP / Vista / 7:" contains a table of download links.

Please select a setup package depending on your platform:

- **Windows 2000/XP/Vista/7/8**
- Linux 32-bit
- Linux 64-bit
- Mac OS X


**NOTE:** There are also more recent *nightly builds* available in the **forums** or (for Debian and Fedora users) in **Jens' Debian repository** and **Jens' Fedora repository**. Please note that we consider nightly builds to be *stable*, usually.

**MIRRORS:** BerliOS mirrors all files *usually* at SourceForge using a "BerliOS robot" **here**. As this sometimes doesn't work, we have mirrored all file releases at SourceForge, too **here**. The latter is managed by us.

**IMPORTANT NOTE:** If you try to download from BerliOS and get a "Too many clients!" - error, you should retry to download the file. According to a BerliOS - admin, this can happen several times, before the download starts. Alternatively use one of the mirrors.

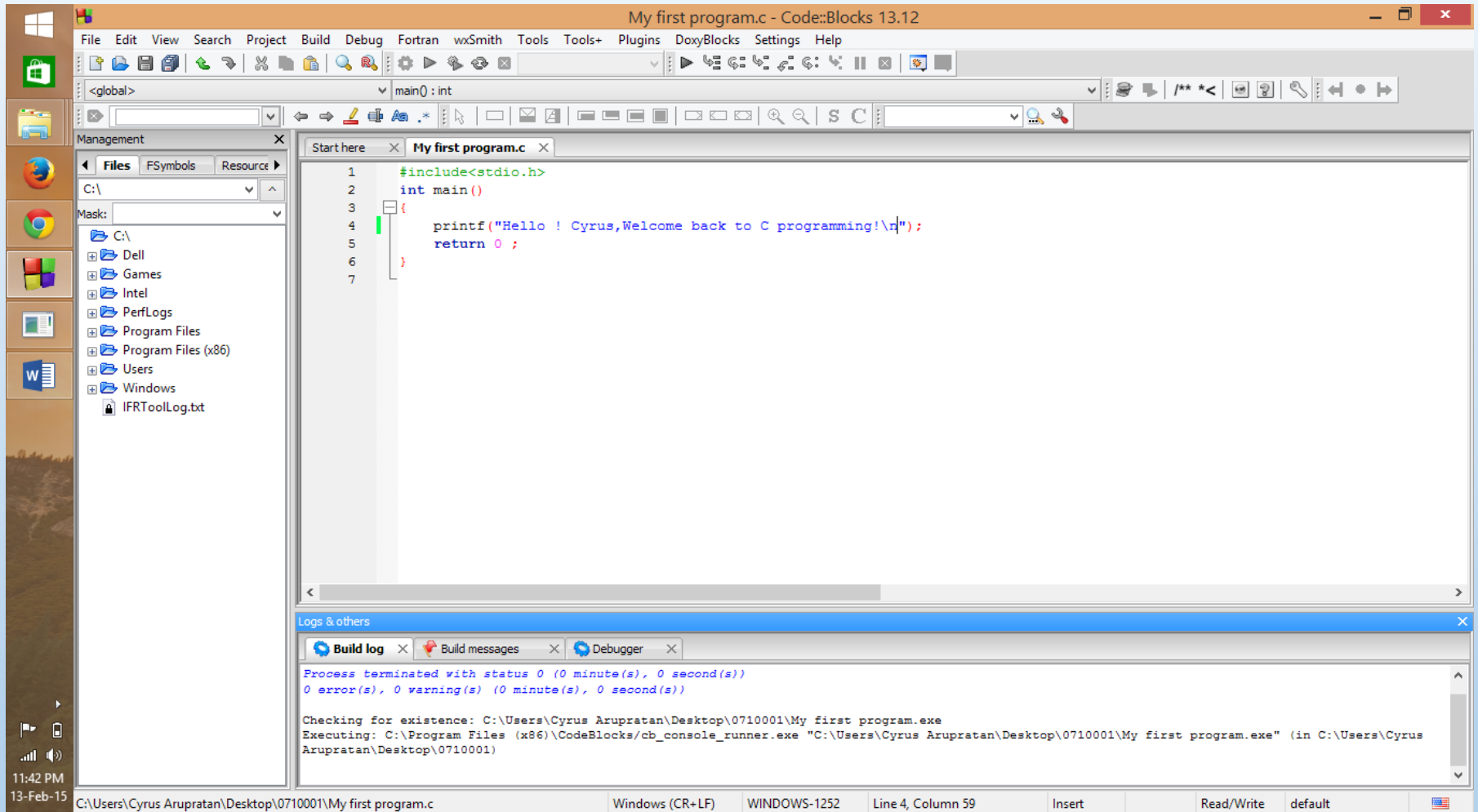
**NOTE:** We have a **Changelog** for 13.12, that gives you an overview over the enhancements and fixes we have put in the new release.

---

 **Windows 2000 / XP / Vista / 7:**

File	Date	Download from
codeblocks-13.12-setup.exe	27 Dec 2013	BerliOS or Sourceforge.net
<b>codeblocks-13.12mingw-setup.exe</b>	27 Dec 2013	BerliOS or <b>Sourceforge.net</b>
codeblocks-13.12mingw-setup-TDM-GCC-481.exe	27 Dec 2013	BerliOS or Sourceforge.net

# Code::Blocks 13.12

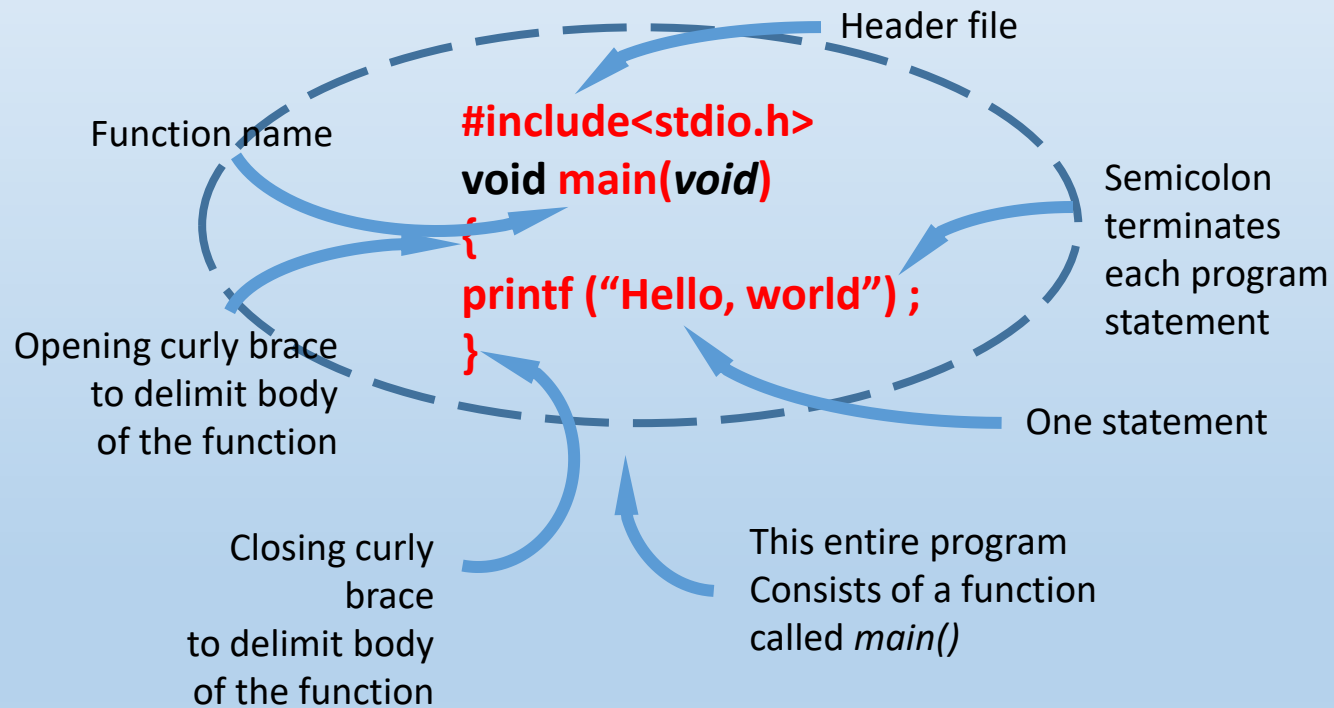


# Basic Structure of A typical C Program

- Documentation Section
- Link Section
- Definition Section
- Global Declaration Section
- main() Function Section
- { Declaration part
- Executable part
- }
- Subprogram section



# A Simple C Program: Example 1



# printf() function:

a useful function from the **standard library** of functions that are accessible by C programs

- The constants on the right are plugged in according to the **Format Specifiers** in the string on the left



The diagram illustrates the mapping between format specifiers and constants in a printf statement. It shows a blue arrow pointing from the '%s' specifier to the string 'Venus', and a white arrow pointing from the '%d' specifier to the number '67'. A dark blue arrow points from the 'million miles' text to the '67' constant, and a grey arrow points from the '\n' specifier to the '67' constant.

```
printf(" %s is %d million miles \n from the sun.", "Venus", 67);
```

- The resulting string is displayed on the monitor

# Example # 2a

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

```
void main(void)
```

```
{
```

```
printf(“ %s is %d million miles away from the sun.”, “Venus”, 67);
```

```
}
```

# Example # 2b

```
# include <stdio.h>
void main(void)
{
printf(“ %s is %d million miles away \n from the sun.”, “Venus”, 67);
}
```

What is the difference between the two codes?

# Escape Characters

## Escape Sequence

\b

\n

\t

\\

\'

\"

## Character Value

Blank space

New line

Tab

Backslash

Apostrophe

Double quote

# Example # 3a

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

```
void main(void)
```

```
{
```

```
int event = 5;
```

```
char heat = 'A';
```

```
float time = 27.25;
```

```
printf (“ \n The winning time in heat %c ”, heat) ;
```

```
printf (“ of event %d was %f.” , event, time);
```

```
}
```

# Variables

- Consists of letters and digits, in any order
- 1st character **must be** letter or underscore, after that you can use numbers.
- ( \_ ) can be considered as a letter
- Both upper- and lowercase are permitted(Case sensitive)
- **Keywords** are not allowed  
(int,char,float,if ,else,void,while signed,const,break,do,return etc.)
- C recognizes only the 1st 31 characters

\_0123456789012345678901234567890

\_0123456789012345678901234567890

they are duplicate

# Variable declaration

- General form

type variable-name;

- Example:

```
int i;
```

```
float p, q, r;
```

```
char a;
```



# Test: Variable name

First\_tag

Valid

Valid?

Not Valid?

char

**Not Valid**

Valid  
Keyword

Not valid?

Price\$

Not valid

Valid?  
Illegal \$

sign  
Not Valid?

group one

Not valid

Valid?  
No Blank

Not Valid?  
space

int\_type

Valid

Valid?  
Keyword, a part of name

Not  
valid?

# Review

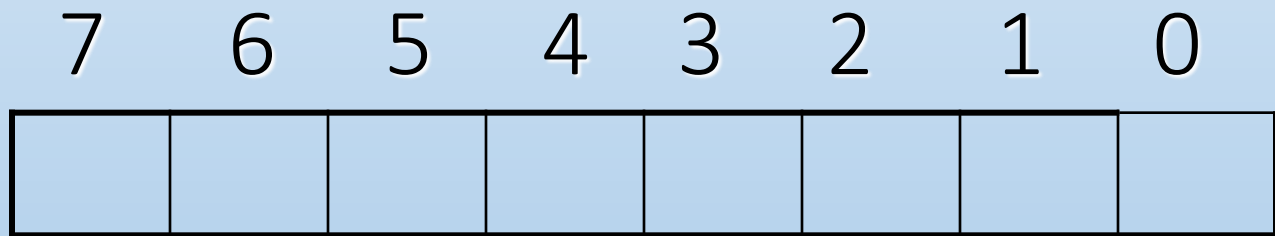
Write a program that will display the following line

“The use of Mobile phones/pen drives is strictly prohibited during the class time”

TIME: 3 MINUTES

# Bits and bytes

- Each piece of information stored within computer's memory is encoded as some **unique combination of zeroes and ones**.
- These 0/1 are called bits.  
1 byte = 8 bits.



# Data types

unsigned char	8 bits	0 to 255
char	8 bits	-128 to 127
enum	16 bits	-32,768 to 32,767
unsigned int	16 bits	0 to 65,535
short int	16 bits	-32,768 to 32,767
int	16 bits	-32,768 to 32,767
unsigned long	32 bits	0 to 4,294,967,295
long	32 bits	-2,147,483,648 to 2,147,483,647
float	32 bits	$3.4 * (10^{-38})$ to $3.4 * (10^{+38})$
double	64 bits	$1.7 * (10^{-308})$ to $1.7 * (10^{+308})$
long double	80 bits	$3.4 * (10^{-4932})$ to $1.1 * (10^{+4932})$

# Write the following program

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

```
void main()
```

```
{
```

```
printf("integer type data takes %d byte",sizeof(int));
```

```
}
```

Try the same for:

float

char

double

# Example for variable size understanding

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

```
Void main()
```

```
{
```

```
int a = 32769;
```

```
printf(“%d”,a);
```

```
}
```

# Format specifiers

% d	Integer	Signed decimal integer
% i	Integer	Signed decimal integer
% o	Integer	Unsigned octal integer
% u	Integer	Unsigned decimal integer
% x	Integer	Unsigned hexadecimal int (with a, b, c, d, e, f)
% X	Integer	Unsigned hexadecimal int (with A, B, C, D, E, F)
% f	Floating point	Signed value of the form [-]dddd.dddd.
% e	Floating point	Signed value of the form [-]d.dddd or e[+/-]ddd
% g	Floating point	Signed value in either e or f form, based on given value and precision. Trailing zeros and the decimal point are printed if necessary.
% E	Floating point	Same as e; with E for exponent.
% G	Floating point	Same as g; with E for exponent if e format used
% c	Character	Single character
% s	String pointer	Prints characters until a null-terminator is pressed or precision is reached
% %	None	Prints the % character

# Format modifiers

Output of Integer Numbers							<b>% wd</b>
Format	Output						
<code>printf(“%d”, 9876);</code>	9	8	7	6			
<code>printf(“%6d”, 9876);</code>	00		9	8	7	6	
<code>printf(“%2d”, 9876);</code>	9	8	7	6			
<code>printf(“%-6d”, 9876);</code>	9	8	7	6			
<code>printf(“%06d”, 9876);</code>	0	0	9	8	7	6	



# Format modifiers

Output of Real Numbers	% w.p f		% w.p e									
Format (y = 98.7654)	Output											
printf(“%7.4f”, y);	9	8	.	7	6	5	4					
printf(“%7.2f”, y);			9	8	.	7	7					
printf(“%-7.2f”, y);	9	8	.	7	7							
printf(“%f”, y);	9	8	.	7	6	5	4					
printf(“%10.2e”, y);			9	.	8	8	e	+	0	1		
printf(“%11.4e”, -y);	-	9	.	8	7	6	5	e	+	0	1	
printf(“%-10.2e”, y);	9	.	8	8	e	+	0	1				
printf(“%e”, y);	9	.	8	7	6	5	4	0	e	+	0	1

# Operators

## Arithmetic operators

C supports all basic arithmetic operations. The operators are –

<b>Operator</b>	<b>Name</b>	<b>Example</b>	<b>Example Result</b>
+	Addition	11 + 51	62
-	Subtraction	34 - 27	7
/	Division	10/3	3.333333333
*	Multiplication	10*3	30
%	Modulus	10%3	1

- $a\%b$  returns the REMAINDER that occurs after performing  $a/b$ . For this operator,  $a$  and  $b$  MUST be integers.
- $10/3 = 3$ ;  $10.0/3 = ?$ ;  $10/3.0 = ?$ ;  $10.0/3.0 = ?$

# *scanf()*

- *scanf()* function allows to accept input from standard in, generally the keyboard
- General form

*scanf("format\_specifier", &variable);*

- “&variable” means address of the variable

*int age;*

*scanf("%d", &age);*

# *scanf()*

## General form

```
printf ("format string" , variables);  
scanf ("format string" , &variables);
```

```
printf("%d",x);
```

```
printf("%d %f",x,y);
```

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

```
scanf("%d %f", &x, &y);
```

# More example of *scanf()*

```
float gpa;  
scanf("%f", &gpa);
```

```
char grade;  
scanf("%c", &grade);
```

```
double score;  
scanf("%lf", &score);
```

# Practice Example

```
#include <stdio.h>

void main()
{
int x=0, y=0;
x = 10;
scanf("%d", &y);
x = x + y;
printf("sum: %d",x);
}
```

## Practice Example

What is the area and perimeter of a circle with a radius of 45 mm?

# Practice Example

```
#include <stdio.h>
void main(void)
{
    int r=45;
    float x, y;
    x= 3.14*r*r; //AREA
    y= 2*3.14*r; //PERIMETER
    printf("Answer:%f and %f",x,y);
}
```



# Practice Example

- Write a C program that will take your **roll number** and **gpa** input and display the information on the monitor as following format

Name: JAMES BOND

Roll No.: 007

GPA: 3.99

# Code for previous Exercise

```
#include <stdio.h>
void main (void)
{
int roll;
float cgpa;
scanf("%d %f",&roll,&cgpa);
printf("Name:\tJahidulHaque\nRoll:\t%d\nCGPA:\t%4.2f",roll,cgpa);
}
```

# Summary of Today's Lesson

- Every C program requires a **main()** function (Use of more than one **main()** is illegal).
- The execution of a function begins at the opening brace ( { ) of the function and ends at the corresponding closing brace ( } ).
- C programs are written in lowercase letters. However, uppercase letters are used for symbolic names and output strings.
- Every program statement in a C program must end with a semicolon.

# Summary of Today's Lesson

- All variables must be declared for their types before they are used in the program.
- Variable must be declared before function calling.
- All the words in a program line must be separated from each other by at least one space, or a tab, or a punctuation mark.
- We must make sure to include *header files* using **#include** directive when the program refers to special names and functions that it does not define.
- Compiler directives such as **define** and **include** are special instructions to the compiler to help it compile a program. They do not end with a semicolon.

# ASSIGNMENT

**SUBMISSION DATE: BEFORE NEXT CLASS**

**SUBMIT BOTH SOFT AND HARD COPY**

# ASSIGNMENT

[1] Write a Program to find the **Perimeter of a Circle**

[Note : **radius** should be scanned from the keyboard.]

[2] Write a program to **compute average of four user given numbers**  
(numbers can be integer or floating types)

## Instructions

- Take care about the structures
- Declare and initialize variables (float/int, x,y)
- Read the input variables
- Write the expression for calculating
- Print the result

# Thank you

*Wit beyond measure is man's greatest treasure*

*-Rowena Ravenclaw*