ME 172

Computer Programming Language Sessional

Dr. Md Mamun
Professor, Dept. of ME, BUET
Mantaka Taimullah
Lecturer, Dept. of ME,BUET

Courtesy: Dr. Noor Al-Quddus
Cyrus Ashok Arupratan Atis
M Aminul Hoque
Dr. Monjur Morshed
M Jamil Hossain
Sourav Saha
Partha Kumar Das

Grading

Items	Percentage
Attendance	10%
Class Performance/ In-class Exercises	20%
Assignment	20%
Mid-Term Quiz	20%
Final Quiz	30%
Total	100%

Rules

- You must come to the class before starting time.
- Don’t copy assignments directly from others. What you submit MUST be your own work unless it is specified as a group submission
- Must submit your assignment on the due date.
- You must come to the class with prior preparation.

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

  1610001

- 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 17.12 for Windows XP / Vista / 7 / 8.x / 10
- 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

CppDroid - C/C++ IDE

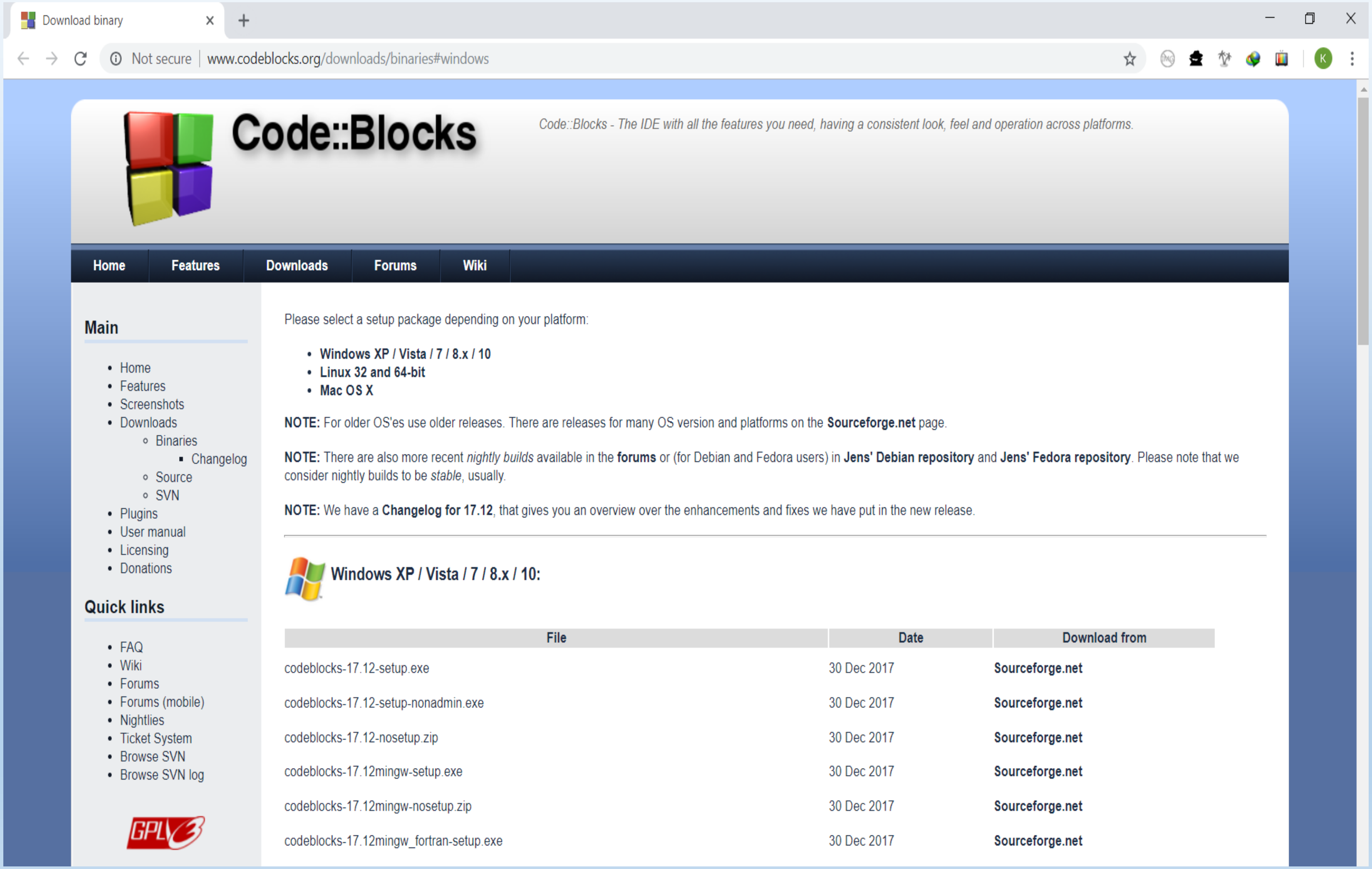
Anton Smirnov - November 20, 2014

Education

Offers in-app purchases

★★★★☆ (10,684)

Downloading Code::Blocks



Code::Blocks - The IDE with all the features you need, having a consistent look, feel and operation across platforms.

Main

- Home
- Features
- Screenshots
- Downloads
  - Binaries
    - Changelog
  - Source
  - SVN
- Plugins
- User manual
- Licensing
- Donations

Quick links

- FAQ
- Wiki
- Forums
- Forums (mobile)
- Nightlies
- Ticket System
- Browse SVN
- Browse SVN log

Please select a setup package depending on your platform:

- **Windows XP / Vista / 7 / 8.x / 10**
- **Linux 32 and 64-bit**
- **Mac OS X**

NOTE: For older OS'es use older releases. There are releases for many OS version and platforms on the **Sourceforge.net** page.

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.

NOTE: We have a **Changelog for 17.12**, that gives you an overview over the enhancements and fixes we have put in the new release.

Windows XP / Vista / 7 / 8.x / 10:

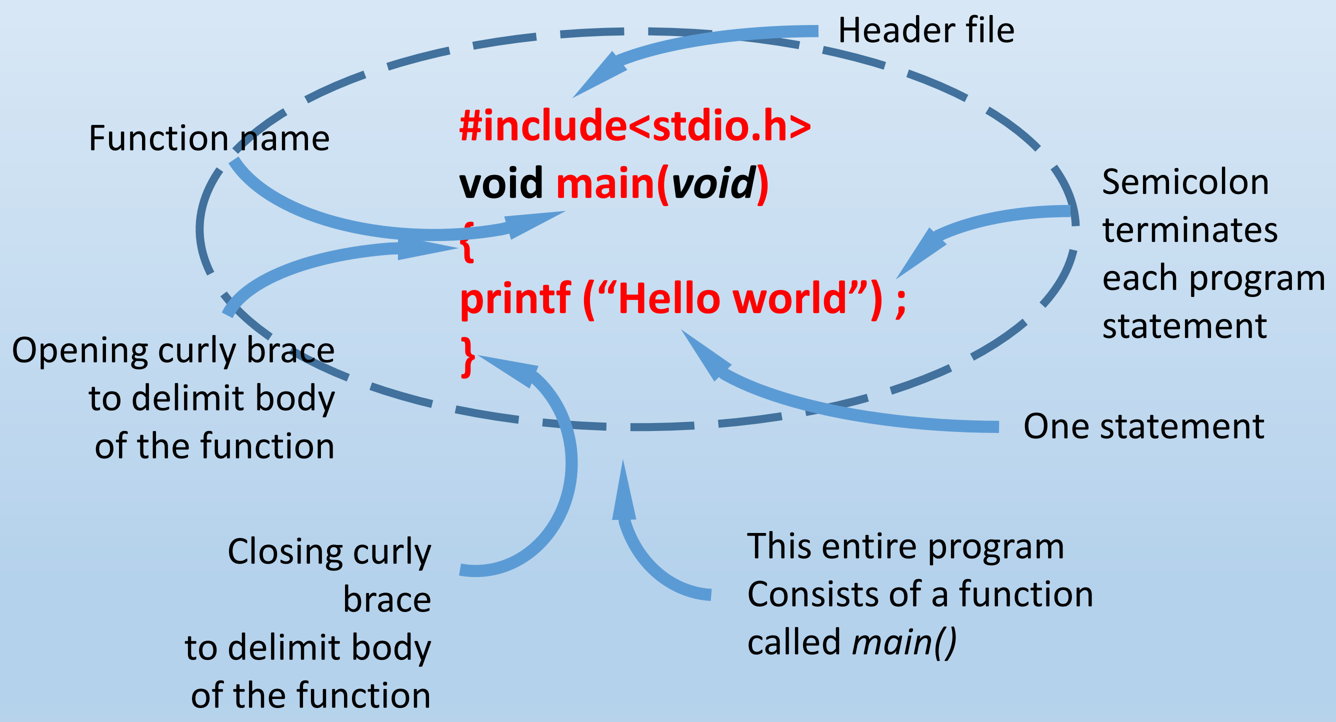
File	Date	Download from
codeblocks-17.12-setup.exe	30 Dec 2017	**Sourceforge.net**
codeblocks-17.12-setup-nonadmin.exe	30 Dec 2017	**Sourceforge.net**
codeblocks-17.12-nosetup.zip	30 Dec 2017	**Sourceforge.net**
codeblocks-17.12mingw-setup.exe	30 Dec 2017	**Sourceforge.net**
codeblocks-17.12mingw-nosetup.zip	30 Dec 2017	**Sourceforge.net**
codeblocks-17.12mingw_fortran-setup.exe	30 Dec 2017	**Sourceforge.net**

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

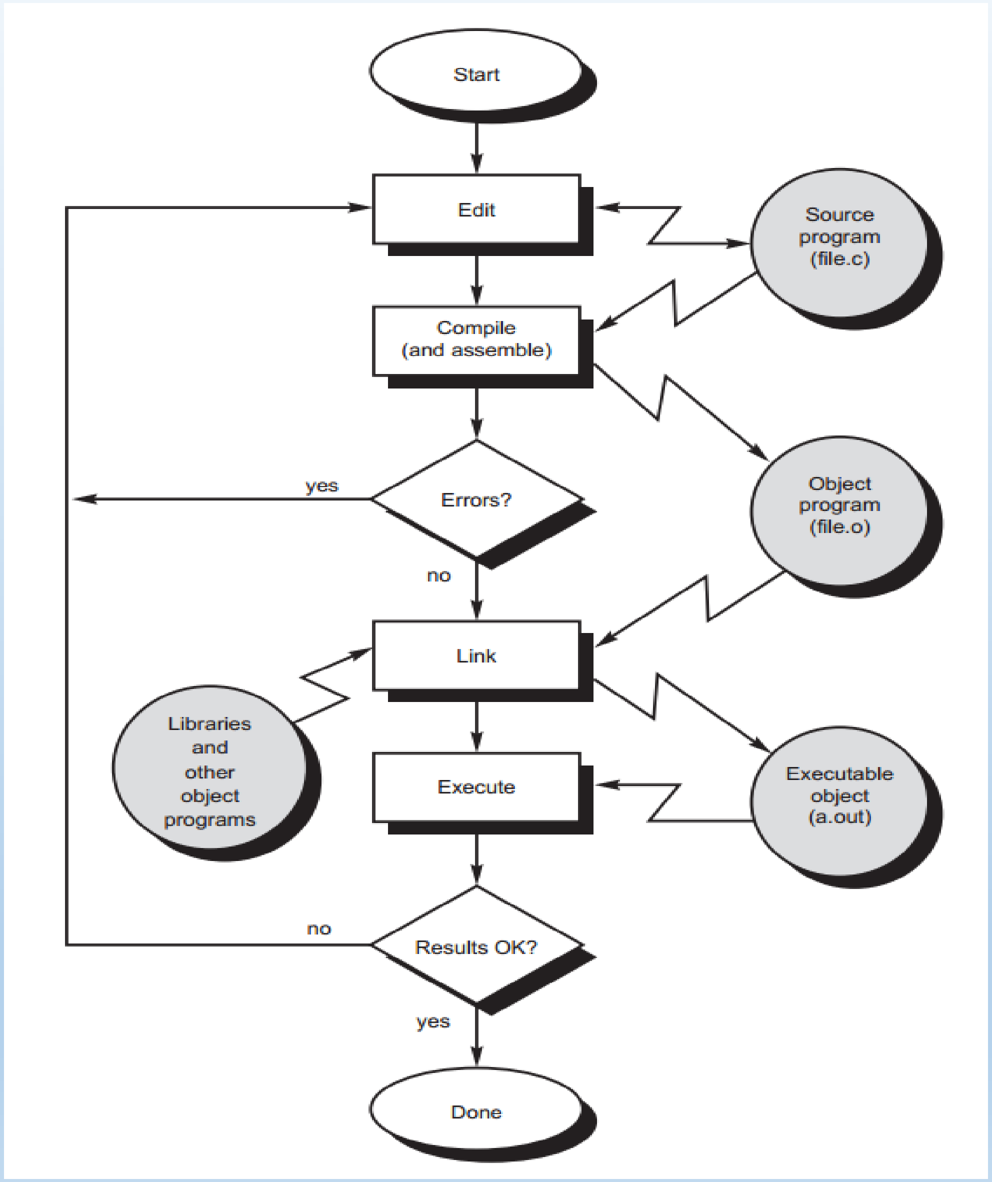
A Simple C Program: Example 1



```
#include<stdio.h>
void main(void)
{
printf ("Hello world") ;
}
```

Header file
Function name
Semicolon terminates each program statement
Opening curly brace to delimit body of the function
One statement
Closing curly brace to delimit body of the function
This entire program Consists of a function called *main()*

Program Flow



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

Keywords and Identifiers

Keywords

are predefined, reserved words used in programming that have special meanings to the compiler. Keywords are part of the syntax and they cannot be used as an identifier.

For example: void, main etc.;

Identifier

refers to name given to entities such as variables, functions, structures etc. Identifiers must be unique. They are created to give a unique name to an entity to identify it during the execution of the program.

Variables and Data types

Variables

In programming, a variable is a container (storage area) to hold data. To indicate the storage area, each variable should be given a unique name (identifier). Variable names are just the symbolic representation of a memory location.

Data types

In C programming, data types are declarations for variables. This determines the type and size of data associated with variables.

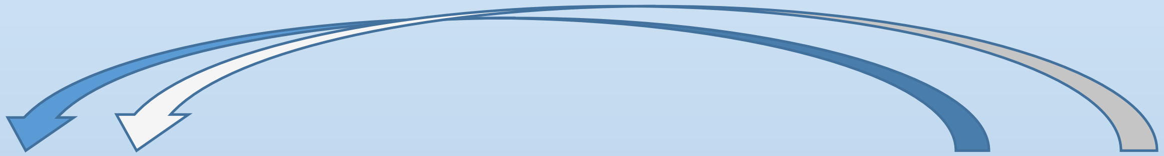
Three basic data types are

i. *int*
ii. *float*
iii. *char*

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



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

- The resulting string is displayed on the monitor

Example # 2a

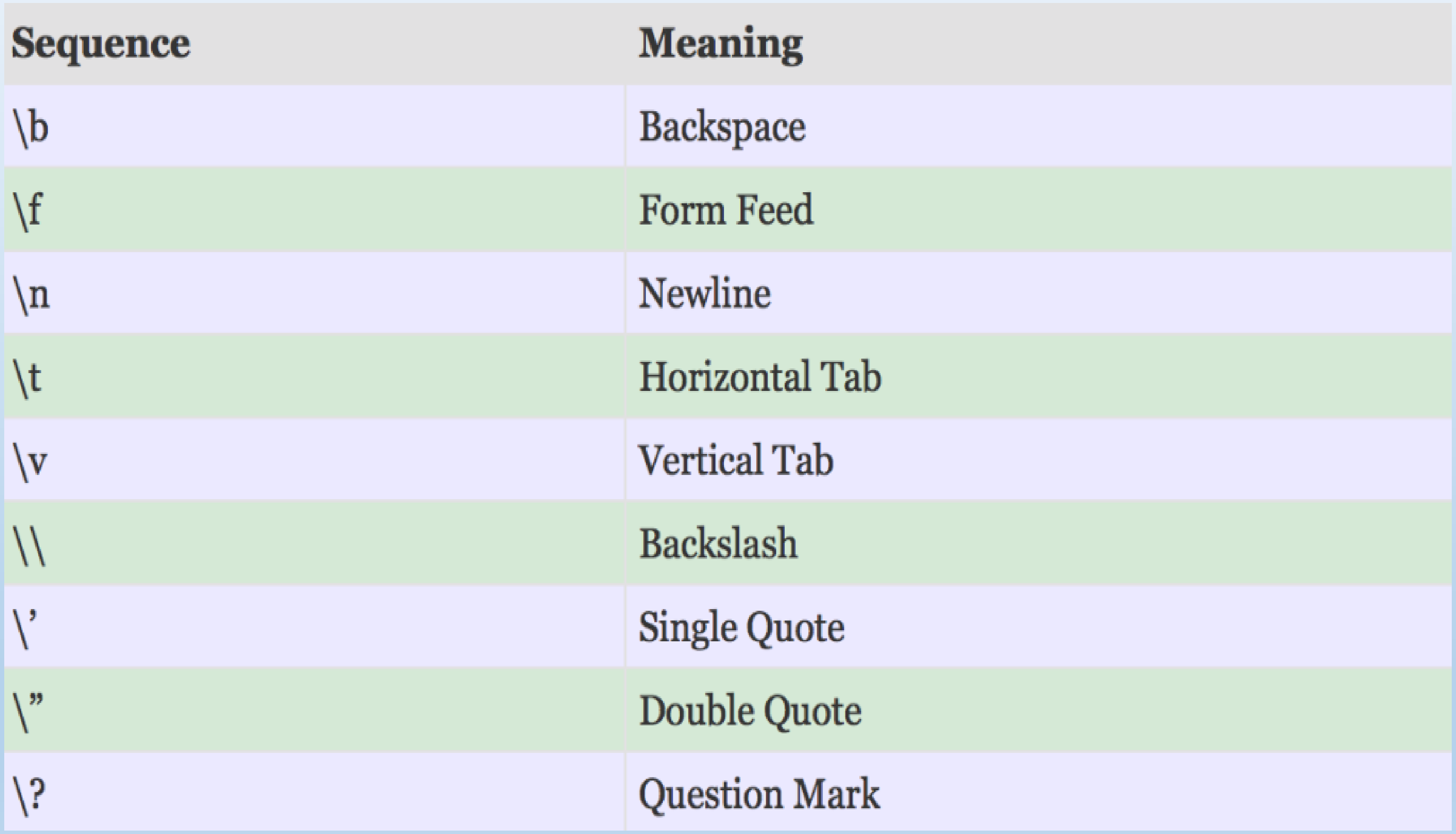
```
# include <stdio.h>
void main(void)
{
printf(" %s is %d million miles away from the sun.", "Venus", 67);
}
```

Example # 2b

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

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

Escape Characters



Sequence	Meaning
\b	Backspace
\f	Form Feed
\n	Newline
\t	Horizontal Tab
\v	Vertical Tab
\\\\	Backslash
\’	Single Quote
\”	Double Quote
\?	Question Mark

Variables

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 seconds.” , event, time);
}
```

Variables

How to name a variable:

- A variable name may consist of letters and digits, in any order
- Underscore (_) can be considered as a letter
- Space can never be considered as a letter for naming a variable
- A variable name must NOT start with a digit. 1st character must be letter or underscore, after that you can use digits.
- Both upper- and lowercase are permitted. (Case sensitive i.e. C recognizes 'a' and 'A' as two different letters.)
- Keywords are not allowed

  (int,char,float,if ,else,void,while signed,const,break,do,return etc.)

Variable declaration

- General form

```
type variable-name;
```

- Example:

```
int i;
float p, q, r;
char  a;
```

Test: Variable name

First_tag
Valid

char
Not Valid
Keyword

Price$
Not valid
Illegal $ sign

group one
Not valid
Blank space is not allowed

intelligent
Valid

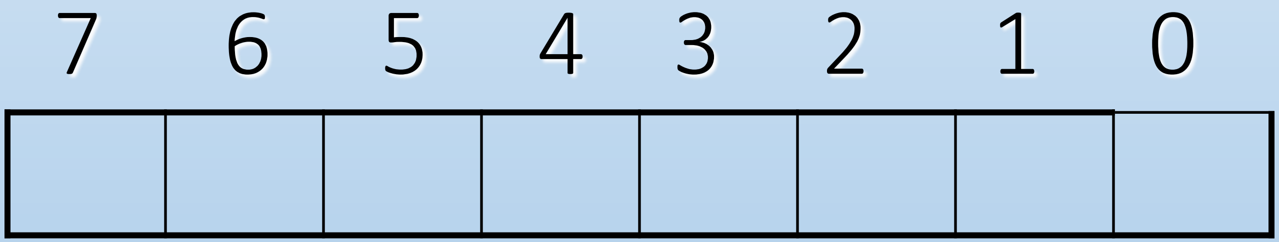
int_type
Valid

Not a keyword, rather keyword is a part of name

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

Type	Storage size	Value range
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	% wd					
Format	Output					
printf("%d", 9876);	9	8	7	6		
printf("%6d", 9876);			9	8	7	6
printf("%2d", 9876);	9	8	7	6		
printf("%-6d", 9876);	9	8	7	6		
printf("%06d", 9876);	0	0	9	8	7	6

width

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 –

Operator	Name	Example	Example Result
+	Addition	11 + 51	62
–	Subtraction	34 – 27	7
/	Division	10/3	3.33333333
*	Multiplication	10*3	30
%	Modulus	10%3	1

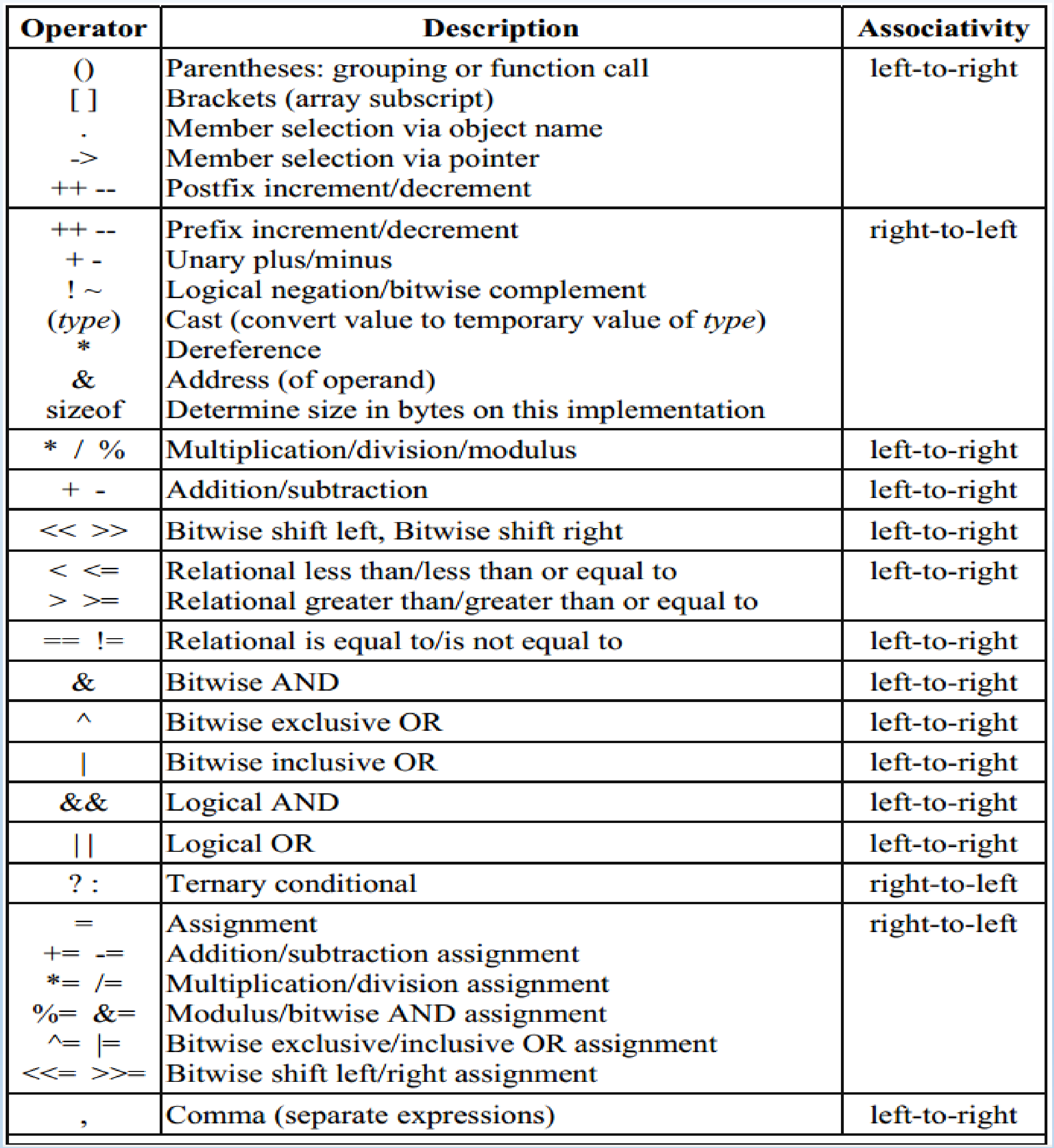
- a%b returns the REMAINDER that occurs after performing a/b. For this operator, a and b MUST be integers.
- 10/3 = 3; 10.0/3 = ?; 10/3.0 = ?; 10.0/3.0 = ?

Arithmetic operators

```
# include <stdio.h>
void main(void)
{
    int  num1,num2,result;
    num1=10;
    num2=3;
    result=num1+num2;
    printf (" %d", result) ;
    result=num1-num2;
    printf (" %d", result) ;
}
```

Change the +/- operator to other arithmetic operators and observe the results

Operator Precedence and Associativity



Operator	Description	Associativity
()	Parentheses: grouping or function call	left-to-right
[ ]	Brackets (array subscript)	
.	Member selection via object name	
->	Member selection via pointer	
++ --	Postfix increment/decrement	
++ --	Prefix increment/decrement	right-to-left
+ -	Unary plus/minus	
! ~	Logical negation/bitwise complement	
(*type*)	Cast (convert value to temporary value of *type*)	
*	Dereference	
&	Address (of operand)	
sizeof	Determine size in bytes on this implementation	
* / %	Multiplication/division/modulus	left-to-right
+ -	Addition/subtraction	left-to-right
<< >>	Bitwise shift left, Bitwise shift right	left-to-right
< <=	Relational less than/less than or equal to	left-to-right
> >=	Relational greater than/greater than or equal to	
== !=	Relational is equal to/is not equal to	left-to-right
&	Bitwise AND	left-to-right
^	Bitwise exclusive OR	left-to-right
|	Bitwise inclusive OR	left-to-right
&&	Logical AND	left-to-right
||	Logical OR	left-to-right
? :	Ternary conditional	right-to-left
=	Assignment	right-to-left
+= -=	Addition/subtraction assignment	
*= /=	Multiplication/division assignment	
%= &=	Modulus/bitwise AND assignment	
^= |=	Bitwise exclusive/inclusive OR assignment	
<<= >>=	Bitwise shift left/right assignment	
,	Comma (separate expressions)	left-to-right

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 area, peri;
  area= 3.1416*r*r;
  peri= 2*3.1416*r;
  printf(“Answer:%f and %f",area,peri);
}
```

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:       Jahidul Haque
Roll No.:   123
GPA:        3.99
```

Code for previous Exercise

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

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 Area 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

Download the slide from http://taimullah.buet.ac.bd/course_materials.html