CS4002
Applied Programming

National University of Computer & Emerging Sciences
Islamabad, Pakistan

Ms. Umarah Qaseem

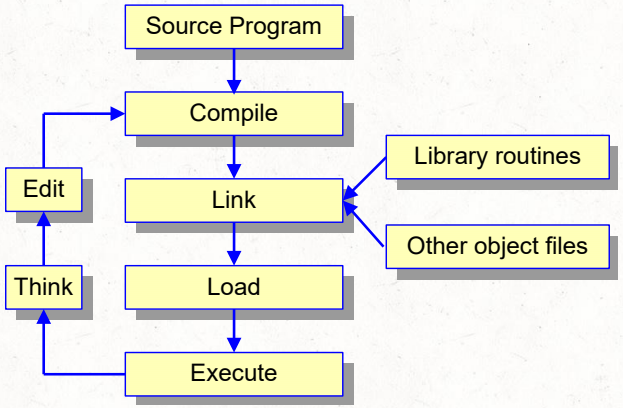
1

Software Development Process

- Editing
- Compiling
- Linking with precompiled files
  - Object files
  - Library modules
- Loading and executing
- Viewing the behavior of the program

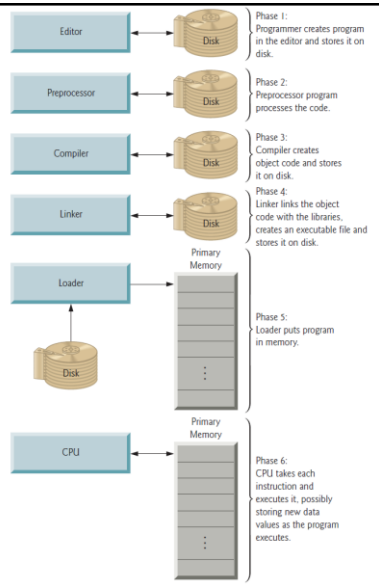
2

Software development activities

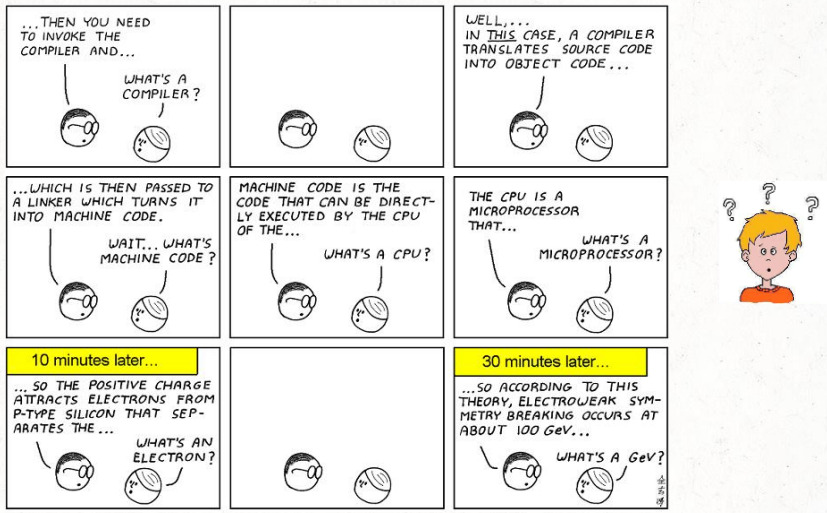


3

Software development activities (Deitel)



4



5

A sample program

```
// my first program in C++

#include <iostream>

void main (void)
     {
            std::cout << "Hello …";
     }
```

6

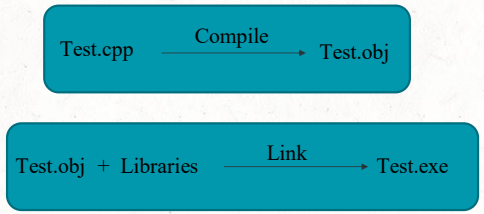
A sample program

```
// my first program in C++

#include <iostream>
Using namespace std;
void main (void)
     {
           cout << "Hello …";
     }
```

7

Code execution

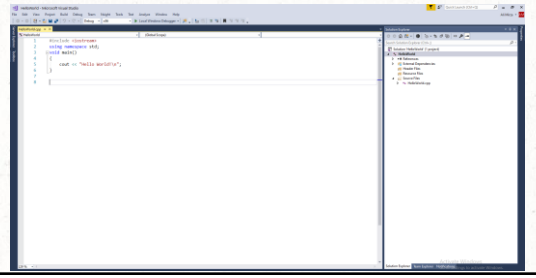
- Source file “Test.cpp”



8

IDEs

- Integrated Development Environments or IDEs
- Supports the entire software development cycle
- E.g., MS Visual C++, Borland
- Provides all the capabilities for developing software
  - Editor
  - Compiler
  - Linker
  - Loader
  - Debugger
  - Viewer



9

Microsoft® Visual Studio®

10

Program errors/ Bugs

- Compiler Errors
- Linker Errors
- Run-time Errors
- Conceptual Errors

11

Program errors

- Compiler Errors

```
// My first C++ program
#include <iostream>
using namespace std;
void main()
{
    cout << "Hello World!\n"
```

Error Test.cpp 4: Statement Missing ; in function main()
Error Test.cpp 4: Compound Statement missing } in function main()

```
// My first C++ program
#include <iostream>
using namespace std;
void main()
{
    cout << "Hello World!\n" ;
}
```

12

Program errors

- Linker Errors

```
// My first C++ program
#include <iostream>
using namespace std;
void Main()
{
    cout << "Hello World!\n" ;
}
```

```
// My first C++ program
#include <iostream>
using namespace std;
void main()
{
    cout << "Hello World!\n" ;
}
```

Linker Error: Undefined Symbol _main

13

Program errors

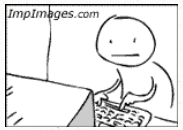
- Run-time Errors

```
#include <iostream>
using namespace std;
void main(void)
{
        int myValue = 0;
        int yourValue = 10;
        double value = yourValue / myValue;
}
```

These errors don't reveal themselves until the program executes.

14

Program errors



- Conceptual Errors

```
#include <iostream>
using namespace std;
void main(void)
{
int value1 = 4;
int value2 = 10;
int value3 = value2 / value1;
cout << "Result =" << value3;
}
```

The program is not doing what you expect it to do.

15

Writing the First Program

16

FIRST PROGRAM IN C++

```
1  // Fig. 2.1: fig02_01.cpp
2  // Text-printing program.
3  #include <iostream>
4
5  // function main be
6  int main()
7  {
8     std::cout << "Welcome to
9
10    return 0; // in
11
12 } // end function main
```

```
Welcome to C++!
```

Single-line comments

Function main returns an

Left brace { begins function body

directive to

Statements end with a semicolon ;

exactly once in every C++ program

Corresponding right brace } ends fu

Stream insertion operator

namespace std

Keyword return is one of several means to exit a function; value 0 indicates that the program terminated successfully

17

FIRST PROGRAM IN C++

- Comments
  - Explain programs to other programmers
  - Improve program readability
  - Ignored by compiler
  - Single-line comment
    - Begin with //
    - Example
      - // This is a text-printing program.
  - Multi-line comment
    - Start with /*
    - End with */

18

FIRST PROGRAM IN C++

- Good programming tip

Every program should begin with a comment that describes the purpose of the program, author, date and time

19

FIRST PROGRAM IN C++

- Preprocessor directives
  - Processed by preprocessor before compiling
  - Begin with #
  - Example
    - #include <iostream>
    - Tells preprocessor to include the input/output stream header file <iostream>
- White space
  - Blank lines, space characters and tabs
  - Used to make programs easier to read
  - Ignored by the compiler

20

FIRST PROGRAM IN C++

- Common Programming Error

Forgetting to include the <iostream> header file in a program that inputs data from the keyboard or outputs data to the screen causes the compiler to issue an error message, because the compiler cannot recognize references to the stream components (e.g., cout).

21

FIRST PROGRAM IN C++

- Function **`main`**
  - A part of every C++ program
  - Exactly one function in a program must be **`main`**
  - Can "**`return`**" a value
  - Example
    - **`int main()`**
    - This **`main`** function returns an integer (whole number)
  - Body is delimited by braces ({})
- Statements
  - Instruct the program to perform an action
  - All statements end with a semicolon (;)

22

FIRST PROGRAM IN C++

- Namespace
  - std::
    - Specifies using a name that belongs to “namespace” std
    - A **namespace** is an abstract container or environment created to hold a logical grouping
- Standard output stream object
  - std::cout
    - “Connected” to screen
    - Defined in input/output stream header file <iostream>

23

FIRST PROGRAM IN C++

- Stream insertion operator <<
  - Value to right (right operand) inserted into left operand
  - Example
    - std::cout << "Hello";
      - Inserts the string "Hello" into the standard output
      - Displays to the screen
- Escape characters
  - A character preceded by "\"
  - Indicates “special” character output
  - Example
    - "\n" Cursor moves to beginning of next line on the screen

24

FIRST PROGRAM IN C

- Common Programming Error

Omitting the semicolon at the end of a C++ statement is a *syntax error*. (Preprocessor directives do not end in a semicolon.) The syntax of a programming language specifies the *rules* for creating a proper program in that language. A syntax error occurs when the compiler encounters code that violates C++'s language rules (i.e., its syntax). The compiler normally issues an *error message* to help the programmer locate and fix the incorrect code. Syntax errors are also called *compiler errors, compile-time errors or compilation errors*, because the compiler detects them during the compilation phase.

25

FIRST PROGRAM IN C++

- return statement
  - One of several means to exit a function
  - When used at the end of **`main`**
    - The value 0 indicates the program terminated successfully
    - Example
      - return 0;

26

FIRST PROGRAM IN C++

- Good programming tip

Many programmers make the last character printed by a function a newline (\n). This ensures that the function will leave the screen cursor positioned at the beginning of a new line. Conventions of this nature encourage software usability—a key goal in software development.

27

FIRST PROGRAM IN C++

- Escape Sequences

Escape sequence	Description
\n	Newline. Position the screen cursor to the beginning of the next line.
\t	Horizontal tab. Move the screen cursor to the next tab stop.
\r	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line.
\a	Alert. Sound the system bell.
\\	Backslash. Used to print a backslash character.
\'	Single quote. Use to print a single quote character.
\"	Double quote. Used to print a double quote character.

28

Modifying Our First C++ Program

- Two examples
  - Print text on one line using multiple statements
    - Each stream insertion resumes printing where the previous one stopped
  - Print text on several lines using a single statement
    - Each newline escape sequence positions the cursor to the beginning of the next line
    - Two newline characters back to back output a blank line

29

Modifying Our First C++ Program

```
// Fig. 2.3: fig02_03.cpp
// Printing a line of text with multiple statements.
#include <iostream> // allows program to output data to the screen

// function main begins program execution
int main()
{
   std::cout << "Welcome ";
   std::cout << "to C++!\n";

   return 0; // indicate that program ended successfully

} // end function main
```

Multiple stream insertion statements produce one line of output

```
Welcome to C++!
```

30

Modifying Our First C++ Program

```
// Fig. 2.4: fig02_04.cpp
// Printing multiple lines of text with a single statement.
#include <iostream> // allows program to output data to the screen

// function main begins program execution
int main()
{
   std::cout << "Welcome\nto\n\nC++!\n";

   return 0; // indicate that program ended successfully

} // end function main
```

Use newline characters to print on multiple lines

```
Welcome
to

C++!
```

31

Concept check

- How to generate the following output:

```
******************
Name
    My name
Enrollment No
    My Enrollment No
******************
```

32

Once a new technology starts rolling, if you're not part of the steamroller, you're part of the road.

--Stewart Brand

33

Variables

34

What is a Variable?

- A variable is a memory address where data can be stored and changed.
- Declaring a variable means specifying both its name and its data type.

35

Declaration

- Declarations
  - Variable Declarations
  - Constant Declarations
- Variable Declarations
  - Syntax

```
<type> <identifier> = <expression>;
```

  - Example

```
int confidenceLevel = 100;
```

36

Declaration

- Constant Declarations
  - Syntax

```
const <type> <identifier> = <expression>;
```

- Example
  - `const double PI = 3.1459;`

37

Declaration

- Variable Declarations
  - Variables are used to store values that can be changed during the program execution
- Syntax:

```
< type > < identifier >;
< type > < identifier > = < expression>;
```

- Examples:

```
int sum;
int total = 3445;
char answer = 'y';
double temperature = -3.14
```

38

Declarations

- A variable has a type and it can contain only values of that type. For example, a variable of the type `int` can only hold integer values
- Variables are not automatically initialized. For example, after declaration

```
int sum;
```

the value of the variable `sum` can be anything (garbage).

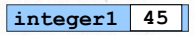
- Thus, it is good practice to initialize variables when they are declared. Once a value has been placed in a variable it stays there until the program deliberately alters it.

39

Declarations

- Constants and variables must be declared before they can be used
- When you declare a constant or a variable, the compiler:
  - Reserves a memory location in which to store the value of the constant or variable.
  - Associates the name of the constant or variable with the memory location.

```
int integer1 = 45;
```



40

Variable Declaration

- All variables must declared **before** use.
  - At the top of the program
  - Just before use.
- Commas are used to separate identifiers of the same type.

```
int count, age;
        or
int count;
int age;
```

- Variables can be initialized to a starting value when they are declared

```
int count = 0;
int age;
```

41

What is an Expression in C++?

- An expression is a valid arrangement of variables, constants, and operators.
- In C++, each expression can be evaluated to compute a value of a given type
- In C++, an expression can be:
  - A variable or a constant (count, 100)
  - An operation (a + b, a * 2)
  - Function call (getRectangleArea(2, 4)) *coming up later in this course*

42

Assignment Operator

- An operator to give (assign) a value to a variable.
- Denote as '='
- Only **variable** can be on the left side.
- An expression is on the right side.
- Variables keep their assigned values until changed by another **assignment statement** or by **reading in** a new value.

43

Assignment Operator Syntax

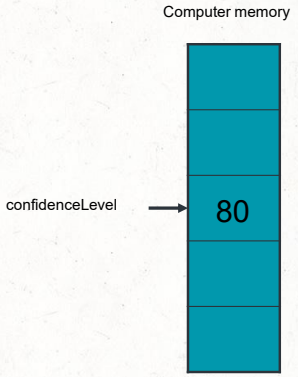
- Variable = Expression
  - First, expression on right is **evaluated**.
  - Then the resulting value is **stored** in the memory location of Variable on left.

NOTE: An automatic type coercion occurs after evaluation but before the value is stored if the types differ for Expression and Variable

44

Variables in C++

- Symbol represents a place to store information
  - Name
  - Value
  - Memory space
- Example: somebody's Confidence Level
  - An integer variable confidenceLevel;
  - confidenceLevel = 80;



45

C++ Data Types

- Integer Data types
  - ✓ char
  - ✓ short
  - ✓ int
  - ✓ long
  - ✓ bool
- Floating Point Data types
  - ✓ float
  - ✓ double
  - ✓ long double

46

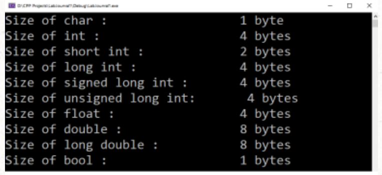
C++ Data types

`sizeof ( )`
returns the size in bytes
`a = sizeof (char)`

Data Type	No. of Bytes	No. of Bits
char	1	8
short	2	16
int	4	32
long	4	32
float	4	32
double	8	64
long double	10	80
bool	1	8

Back to C++

```
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 cout << "Size of char : \t\t\t" << sizeof(char)
        << " byte" << endl;
    cout << "Size of int : \t\t\t" << sizeof(int)
        << " bytes" << endl;
    cout << "Size of short int : \t\t" << sizeof(short int)
        << " bytes" << endl;
    cout << "Size of long int : \t\t" << sizeof(long int)
        << " bytes" << endl;
    cout << "Size of signed long int : \t" << sizeof(signed long int)
        << " bytes" << endl;
    cout << "Size of unsigned long int:\t " << sizeof(unsigned long int)
        << " bytes" << endl;
    cout << "Size of float : \t\t" << sizeof(float)
        << " bytes" << endl;
    cout << "Size of double : \t\t" << sizeof(double)
        << " bytes" << endl;
    cout << "Size of long double : \t\t" << sizeof(long double)
        << " bytes" << endl;
     cout << "Size of bool : \t\t\t" << sizeof(bool)
        << " bytes" << endl;
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```



```
Size of char :                  1 byte
Size of int :                   4 bytes
Size of short int :             2 bytes
Size of long int :              4 bytes
Size of signed long int :       4 bytes
Size of unsigned long int:       4 bytes
Size of float :                 4 bytes
Size of double :                8 bytes
Size of long double :           8 bytes
Size of bool :                  1 bytes
```

C++ Data types

- Integer Data Types
  - Range
  - Unsigned and Signed Data Types

Type	Sign	Size	Min Value	Max Value
short	signed	16 bits	-32768	32767
	unsigned		0	65535
int	signed	32 bits	-2,147,483,648	2,147,483,647
	unsigned		0	4,294,967,295
long	signed	32 bits	-2,147,483,648	2,147,483,647
	unsigned		0	4,294,967,295
bool	-	8 bits	false/0	true/1

49

C++ Data types

- Floating Data Types
  - Range
  - Unsigned and Signed Data Types

Type	Sign	Size	Min Value	Max Value
float	signed	32 bits	-2,147,483,648	2,147,483,647
double	signed	64 bits	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
long double	signed	80 bits	-604,462,909,807,314,587,353,088	604,462,909,807,314,587,353,087

50

Character Data Types

- Represent single characters
- declared as char
- Stored by ASCII values
  - ASCII (American Standard Code for Information Interchange)
- 1 byte for each char

- Byte Values:
  - 00000000

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

= 0

  - 11111111

7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

= 255

52

DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII	DEC	ASCII
1	☺	32	space	64	@	96	`	128	Ç	160	á	192	└	224	Ó
2	☻	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
3	♥	34	"	66	B	98	b	130	è	162	ó	194	┬	226	Ô
4	♦	35	#	67	C	99	c	131	â	163	ú	195	├	227	Ò
5	♣	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	õ
6	♠	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	Õ
7	•	38	&	70	F	102	f	134	å	166	ª	198	ã	230	µ
8	◘	39	'	71	G	103	g	135	ç	167	º	199	Ã	231	þ
9	○	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Þ
10	◙	41	)	73	I	105	i	137	ë	169	®	201	╔	233	Ú
11	♂	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Û
12	♀	43	+	75	K	107	k	139	ï	171	½	203	╦	235	Ù
13	♪	44	,	76	L	108	l	140	î	172	¼	204	╠	236	ý
14	♫	45	-	77	M	109	m	141	ì	173	¡	205	═	237	Ý
15	☼	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	¯
16	►	47	/	79	O	111	o	143	Å	175	»	207	¤	239	´
17	◄	48	0	80	P	112	p	144	È	176	░	208	ð	240	-
18	↕	49	1	81	Q	113	q	145	æ	177	▒	209	Ð	241	±
19	‼	50	2	82	R	114	r	146	Æ	178	▓	210	Ê	242	‗
20	¶	51	3	83	S	115	s	147	ô	179	│	211	Ë	243	¾
21	§	52	4	84	T	116	t	148	ö	180	┤	212	È	244	¶
22	▬	53	5	85	U	117	u	149	ò	181	Á	213	ı	245	§
23	↨	54	6	86	V	118	v	150	û	182	Â	214	Í	246	÷
24	↑	55	7	87	W	119	t	151	ù	183	À	215	Î	247	¸
25	↓	56	8	88	X	120	x	152	ÿ	184	©	216	Ï	248	°
26	→	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	¨
27	←	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
28	∟	59	;	91	[	123	{	155	ø	187	╗	219	█	251	¹
29	↔	60	<	92	\	124	|	156	£	188	╝	220	▄	252	³
30	▲	61	=	93	]	125	}	157	Ø	189	¢	221	¦	253	²
31	▼	62	>	94	^	126	~	158	×	190	¥	222	Ì	254	■
		63	?	95	_	127	⌂	159	ƒ	191	┐	223	▀	255	space

53

Identifiers

- Names of variables
- Series of Characters (letters, digits, underscores)
- Must NOT start with a digit (0 – 9)
- Must not be a C++ keyword
- Case Sensitive
- Exercise
  - Which of these are valid identifiers
    - `floating`
    - `int`
    - `Int`
    - `main3`
    - `4yi`

54

Lvalues and Rvalues

- Lvalues
  - Expressions that can appear on left side of equation
  - Can be changed (i.e., variables)
    - `x = 4;`
- Rvalues
  - Only appear on right side of equation
  - Constants, such as numbers
  - Cannot write `4 = x;`

Lvalues can be used as Rvalues, but not vice versa

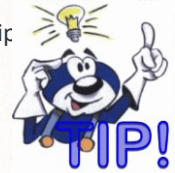
55

Example (Add two numbers)

```
        // Example
    // Addition program.
    #include <iostream>
        using namespace std;
    // function main begins program execution
    void main()
    {
      int integer1;  // first number to be input by user
      int integer2;  // second number to be input by user
      int sum;       // variable in which sum will be stored

      cout << "Enter first integer:\n";  // prompt
      cin >> integer1;                   // read an integer

      cout << "Enter second integer:\n"; // prompt
      cin >> integer2;                   // read an integer

      sum = integer1 + integer2;  // assign result to sum

      cout << "Sum is " << sum << endl; // print sum



   } // end function main
```

56

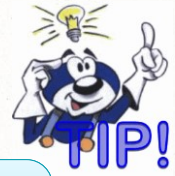
Good programming tip



Some programmers prefer to declare each variable on a separate line. This format allows for easy insertion of a descriptive comment next to each declaration.

57

Good programming tips



Choosing meaningful identifiers helps make a program self-documenting—a person can understand the program simply by reading it rather than having to refer to manuals or comments.

Avoid using abbreviations in identifiers. This promotes program readability.

58

Portability Tip

C++ allows identifiers of any length, but your C++ implementation may impose some restrictions on the length of identifiers. Use identifiers of 31 characters or fewer to ensure portability.

59

Input value

- Input Source?
  - Console
  - Data File
- Console Input
  - cin >>
  - In the header file “iostream”

60

Escape Sequences

1. **\n** (New line) – We use it to shift the cursor control to the new line
2. **\t** (Horizontal tab) – We use it to shift the cursor to a couple of spaces to the right in the same line.
3. **\a** (Audible bell) – A beep is generated indicating the execution of the program to alert the user.
4. **\r** (Carriage Return) – We use it to position the cursor to the beginning of the current line.
5. **** (Backslash) – We use it to display the backslash character.
6. **\’** (Apostrophe or single quotation mark) – We use it to display the single-quotation mark.
7. **\”** (Double quotation mark)- We use it to display the double-quotation mark.
8. **\0** (Null character) – We use it to represent the termination of the string.
9. **\?** (Question mark) – We use it to display the question mark. (?)
10. **\nnn** (Octal number)- We use it to represent an octal number.
11. **\xhh** (Hexadecimal number) – We use it to represent a hexadecimal number.
12. **\v** (Vertical tab)
13. **\b** (Backspace)
14. **\e** (Escape character)
15. **\f** (Form Feed page break)

61