Southern Technical University

Basra Technical Institute

Department of Computer Networks and Software Technologies

الجامعة التقنية الجنوبية

المعهد التقني التكنولوجي – البصرة

قسم تقنيات شبكات وبرامجيات الحاسوب

حقيبة تعليمية

لمادة

البرمجة بلغة

C++/1

لطلبة المرحلة الأولى

العام الدراسي 2024_2025

إعداد مدرسة المادة

م. م. سحر سامي فاضل

Southern Technical University

Basra Technical Institute

Department of Computer Networks and Software Technologies

الجامعة التقنية الجنوبية

المعهد التقني التكنولوجي – البصرة

قسم تقنيات شبكات وبرامجيات الحاسوب

Learning package

Programming with C++ Language

C++/1

For

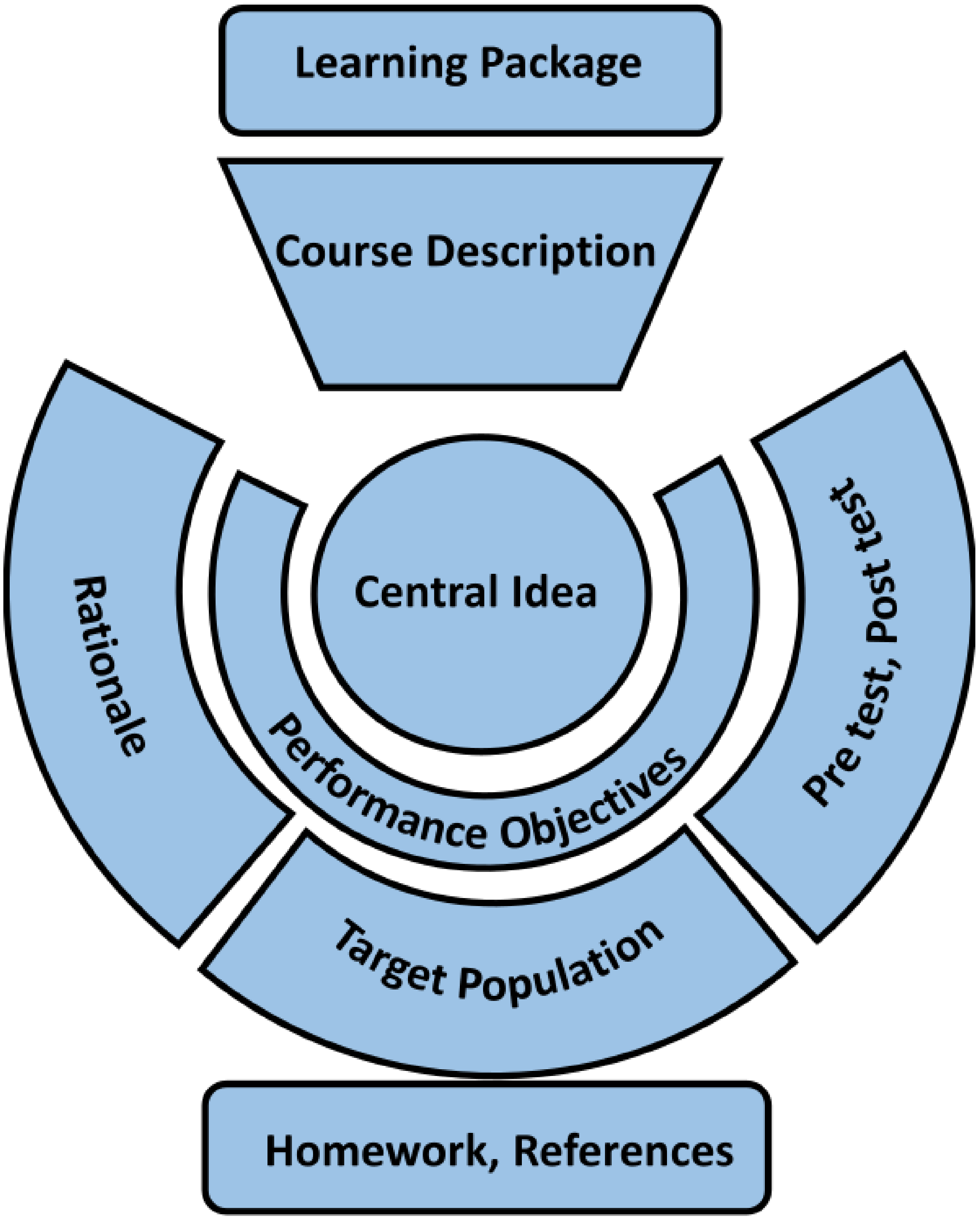
First year student

2025-2026

By

Sahar Sami Fadhil

Assistant Lectuerer


Learning Package
Course Description
Central Idea
Rationale
Performance Objectives
Pre test, Post test
Target Population
Homework, References

Course Description Form

1. Course Name:
Programming with C++ language/1
2. Course Code:
3. Semester / Year:
semester
4. Description Preparation Date:
25/6/2025
5. Available Attendance Forms:
Attendance only
6. Number of Credit Hours (Total) / Number of Units (Total)
60 hours (theoretical + practical) at a rate of 4 hours per week (2 theoretical practical)
7. Course administrator's name (mention all, if more than one name)
Name: Sahar Sami Fadhil Email: sahar.fadhil@stu.edu.iq
8. Course Objectives
The aim of this course is to learn programming in C++.
9. Teaching and Learning Strategies
In-person lectures, short tests, assignments and practical application in the laboratory

10. Course Structure					
Week	**Hours**	**Required Learning Outcomes**	**Unit or subject name**	**Learning method**	**Evaluation method**
1&2	**2 hours Theoretical + 2 hours Practical**	The student is abl to master the cra of programming in general and programming in C++ in particular.	Abstract of programming languages ● What's a program and what's a programming language ● The compiler and the interpreter ● Levels of programming languages ● C++ language : beginning, development, its location within levels of programming languages	1-Explaining The scientific material 2-Asking questions related to the material	Weekly And Daily Written exams, mid-term exam and end-of-semester exam
3			Basic essentials for C++ language/ C++ language concepts ● What's C++ program contains? ● What are the basic files? Simple explanation for basic files, that C++ program include		
4&5			Basic element and tools of C++ language		

			• Language symbols • Definitions name • keywords • Constant represent • Variables represent		
6			Data types in C++, and they represent methods in memory • char type • integer type • real type • Boolean (logical) type		
7			Expressions types in C++ language, how formulate expression: • Arithmetic expression /deferent arithmetic operation and its priorities / conversion manner of arithmetic expression to Arithmetic expression in C++ language/deferent examples		

8&9			• Relational expression/ relational operations and its priorities/ formulate Relational expression • Logical expression/ logical operation and its priorities/formulate Logical expression • Compound expression/ priorities table of public operations/ deferent examples		
10			• Spaces and brackets • Type of comments • Increment/ decrement Operators		
11			Assignment statement, its types/ with explanation examples • Arithmetic expression (equation) • deferent images for equations belong to C++		

			language		
12			Comma operator and conditional operator		
13 & 14			Control, conditional, and loop statements • cond. Statement ○ If conditional statement ○ If-else conditional statement ○ Nested if-else conditional statement		
15			• switch conditional statement		

11. Course Evaluation

The distribution is as follows:

20 points for the theoretical mid-term exam
20 points = the practical = = =
10 points for student activity during the semester
Total 50 points for the annual effort

50 points for the final exam
40 points for the theoretical exam at the end of the semester
10 points for the practical exam at the end of the semester

12. Learning and Teaching Resources

E−books and topics found on scientific and software websites on the Internet

Southern Technical University

Basra Technical Institute

Department of Computer Networks and Software Technologies

الجامعة التقنية الجنوبية

المعهد التقني التكنولوجي – البصرة

قسم تقنيات شبكات وبرامجيات الحاسوب

Learning package

Abstract of Programming Languages

For

First year student

2025-2026

By

Sahar Sami Fadhil

Assistant Lectuerer

Weeks (1&2)

Pre-Question

- **What do you know about programming languages (old and new)?**

Abstract of programming languages

What's a programming language

- A program is a set of instructions following the rules of the chosen language.
- Without programs, computers are useless.
- A program is like a recipe, it contains a list of ingredients (called variables) and a list of directions (called statements) that tell the computer what to do with the variables.
- Programming languages can be used to create computer programs.

A programming language is a set of commands, instructions, and other syntax use to create a software program. Languages that programmers use to write code (or program) are called "high-level languages." This code can be compiled into a "low-level language," which is recognized directly by the computer hardware.

Programs written in a high-level language need to be translated into machine language before they can be executed. There are two ways to do this:

1– Compile the program

- Compile is to transform a program written in a high level programming language from source code into object code.
- This can be done by using a tool called compiler.
- A compiler reads the whole source code and translates it into a complete

machine code program to perform the required tasks which is output as a new file.

2– Interpret the program

- Interpreter is a program that executes instructions written in a high-level language.
- An interpreter reads the source code one instruction or line at a time, converts this line into machine code and executes it.

Levels of programming languages

Low-level languages include assembly and machine languages. An assembly language contains a list of basic instructions and is much more difficult to read than a high-level language. In rare cases, a programmer may decide to code a basic program in an assembly language to ensure it operates as efficiently as possible. An assembler can be used to translate the assembly code into machine code. The machine code, or machine language, contains a series of binary codes that are understood directly by a computer's CPU. Needless to say, machine language is not designed to be human readable.

High-level languages are designed to be easy to read and understand. This allows programmers to write source code in a natural fashion, using logical words and symbols. For example, reserved words like function, while, if, and else are used in most major programming languages. Symbols like <, >, ==, and != are common operators. Many high-level languages are similar enough that programmers can easily understand source code written in multiple languages.

Examples of high-level languages include C++, Java, Perl, and PHP. Languages like C++ and Java are called "compiled languages" since the source code must first be compiled in order to run. Languages like Perl and PHP are called "interpreted languages" since the source code can be run through an interpreter without being compiled. Generally, compiled languages are used to create software applications, while interpreted languages are used for running scripts, such as those used to generate content for dynamic websites.

Types of Programming Languages

There are three types of programming language:

– Machine language (Low-level language)

– Assembly language (Low-level language)

– High-level language

Example:

– Machine language : 10110000 01100001

– Assembly language : mov a1, #061h

– Meaning: Move the hexadecimal value 61 (97 decimal) into the processor register named "a1".

C++ language : beginning, development, its locaton within levels of programming languages

In the early 1970s, Dennis Ritchie of Bell Laboratories (in Murray Hill, New Jersey) was engaged in a project to develop a new operating system. Ritchie discovered that in order to accomplish his task he needed the use of a programming language that was concise and that produced compact and speedy programs. This need led Ritchie to develop the programming language called C.

In the early 1980's, also at Bell Laboratories, another programming language was created which was based upon the C language. This new language was developed by Bjarne Stroustrup and was named (C with Classes) but later it was renamed C++ in 1983. Stroustrup states that the purpose of C++ is to make writing good programs easier and more pleasant for the individual programmer. When he designed C++, he added OOP (Object Oriented Programming) features to C without significantly changing the C component. Thus C++ is a "relative" (called a superset) of C, meaning that any valid C program is also a valid C++ program.

C++ is regarded a **middle-level** language, as it comprises a combination of both high-level and low-level language features.

H.W

- **What is the difference between the compiler and an interpreter?**
- **What is the classification of the C++ language within the programming language levels?**

Resources

- **E–books and topics found on scientific and software websites on the Internet**

Southern Technical University

Basra Technical Institute

Department of Computer Networks and Software Technologies

الجامعة التقنية الجنوبية

المعهد التقني التكنولوجي – البصرة

قسم تقنيات شبكات وبرامجيات الحاسوب

Learning package

Basic Essentials for C++ Language
C++ Language Concepts

For

First year student

2025-2026

By

Sahar Sami Fadhil

Assistant Lectuerer

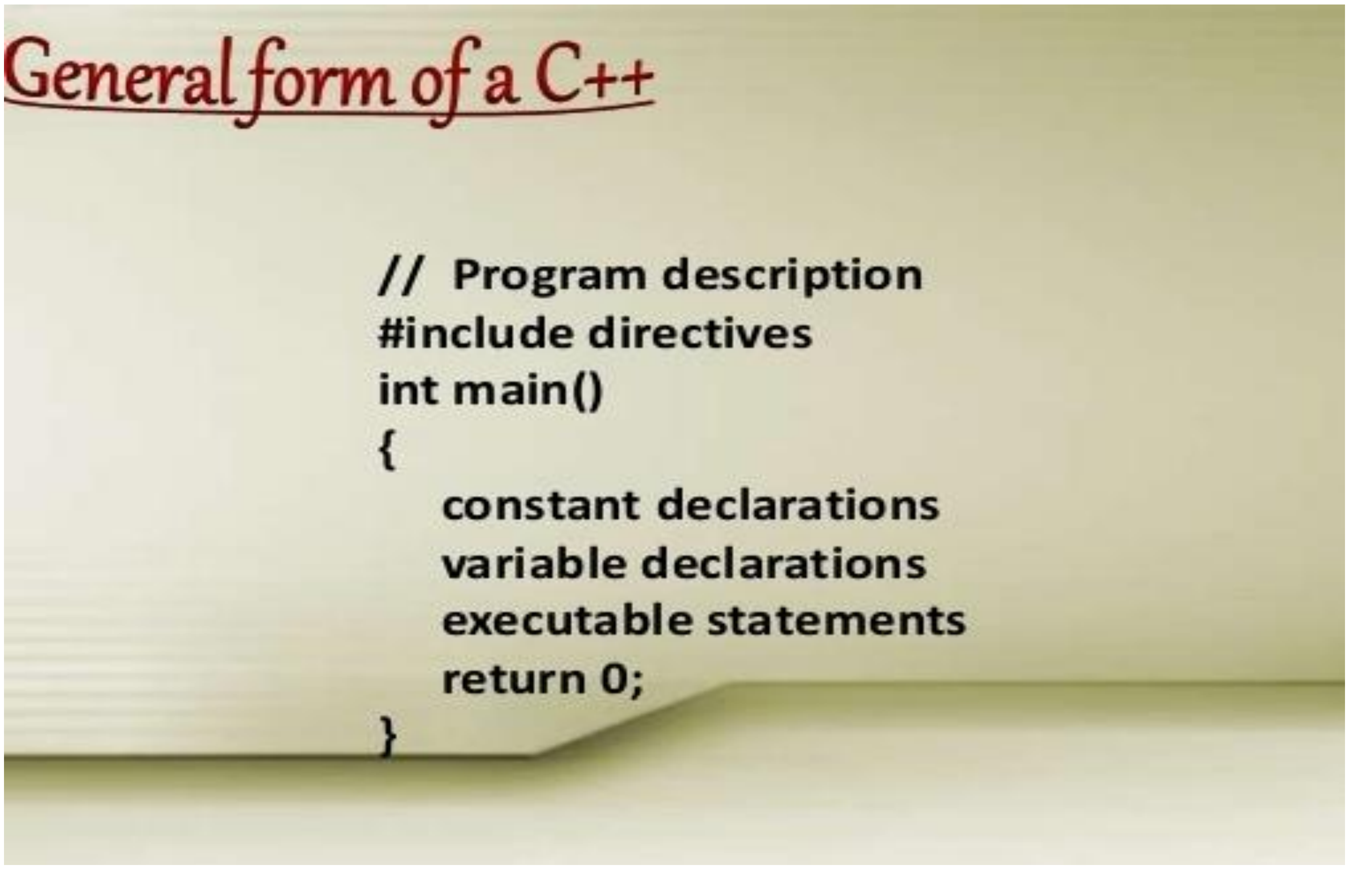
Week (3)

Pre-Question

- Where are programming language commands found (in general)?

Basic essentials for C++ languages/ C++ language concepts

What's C++ program contains?



General form of a C++

```
// Program description
#include directives
int main()
{
    constant declarations
    variable declarations
    executable statements
    return 0;
}
```

What are the basic files? Simple explanation for basic files, that C++ program include

The C++ standard library contains files containing the standard functions that your program may use. These files are known as **header files**.

Header Files in C++

Header files contain definitions of **Functions and Variables**, which is imported or used into any C++ program by using the #include statement. Header file have an extension ".h" which contains C++ function declaration.

Each header file contains information (or declarations) for a particular group of functions. Like **iostream.h** header file contains declarations of standard input and output functions available in C++ which is used for get the input and print the output. Similarly, the header file **math.h** contains declarations of mathematical functions available in C++.

H.W

- **What does the C++ standard library include?**

Resources

- **E–books and topics found on scientific and software websites on the Internet**

Southern Technical University

Basra Technical Institute

Department of Computer Networks and Software Technologies

الجامعة التقنية الجنوبية

المعهد التقني التكنولوجي – البصرة

قسم تقنيات شبكات وبرامجيات الحاسوب

Learning package

Basic Element and Tools of C++ Language

For

First year student

2025-2026

By

Sahar Sami Fadhil

Assistant Lectuerer

Weeks (4&5)

Pre-Question

- What are the basic elements of a natural language and what are the basic elements that correspond to them in a programming language?

Basic elements and tools of C++ language

Language symbols

What is Character Set in C++?

Character set is the combination of English language (Alphabets and White spaces) and math's symbols (Digits and Special symbols). Character Set means that the characters and symbols that a C++ Program can understand and accept. These are grouped to form the commands, expressions, words, C-statements and other tokens for C++ Language.

There are mainly four categories of the character set :

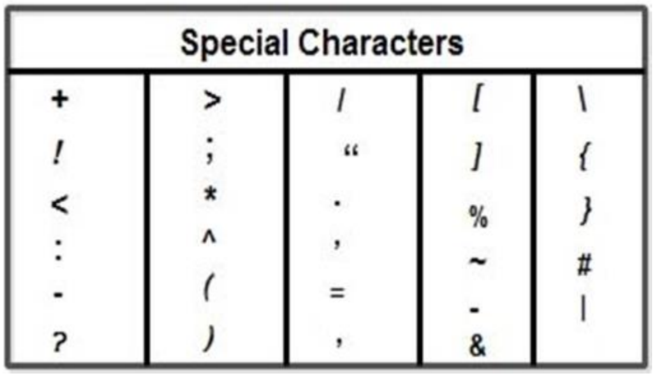
1. Alphabets: Alphabets are represented by A-Z or a-z. C- Language is case sensitive so it takes different meaning for small and upper case letters.

2. Digits: Digits are represented by 0-9 or by combination of these digits.

3. Special Symbols : All the keyboard keys except alphabet, digits and white spaces are the special symbols. These are some punctuation marks and some special symbols used for special purpose.

There are total 30 special symbols used in the C-programming. Special symbols are used for C-statements like to create an arithmetic statement +, -, * etc. , to create relational statement <, >, <=, >=, == etc. , to create assignment statement =, to create logical statement &&, II etc.

The special characters are listed in Table



Special Characters				
+	>	/	[	\
!	;	“	]	{
<	*	.	%	}
:	^	,	~	#
-	(	=	-	|
?	)	,	&	

In addition to these characters, C++ also uses a combination of characters to represent special conditions. For example. character combinations such as '\nt, '\b' and '\t' are used to represent newline, backspace and horizontal tab respectively.

<u>Note</u> *Some tokens made up of two characters (with no blank)--still considered as*

single symbol

4. White Spaces: White spaces has blank space, new line return, Horizontal tab space, carriage ctrl etc. are all used for special purpose. Also note that C Compiler always ignore these white space characters in both high level and low level programming.

Definitions names or (I d e n t i f i e r s)

- Identifiers: used as names for variables, constants, and functions
- Identifiers: consist of letters, digits, and underscore character (_)
- Identifiers: **must begin** with letter or underscore (best not to use underscore)

Legal Identifier Examples: first conversion payRate

Illegal Identifier Examples:

employee Salary (cannot use space)

Hello! (cannot use special characters, like exclamation mark)

one+two (cannot use special characters, like + character)

2nd (cannot begin with digit)

Keywords

Another type of tokens is **Reserved Words** or **Keywords**.

Reserved Words: always lowercase, each considered to be single symbol with special meaning

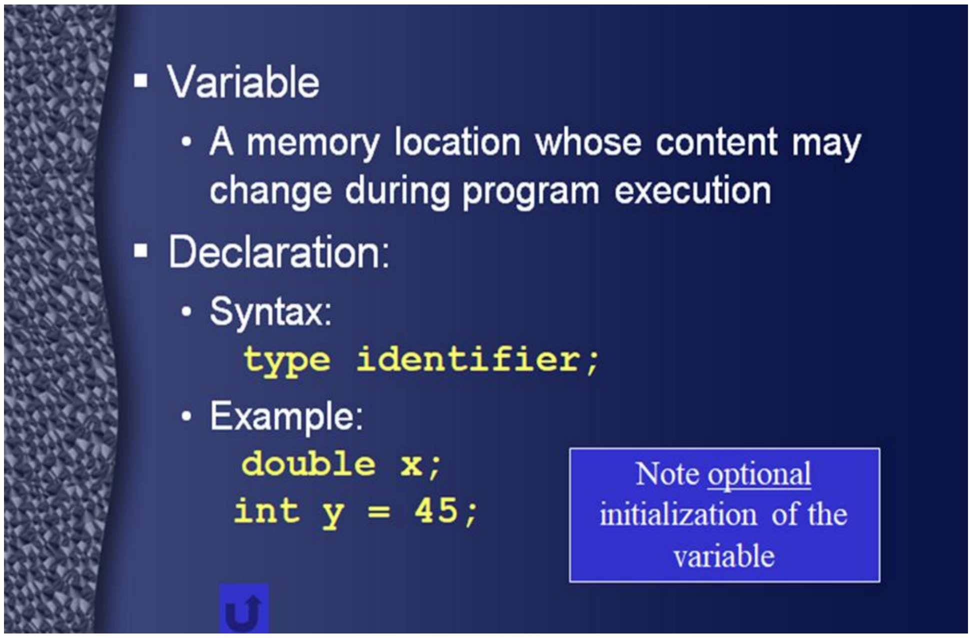
Examples:

```
int     float    double  char    void    return
```

Constant represent & Variables represent

Declaration of Variable

The declaration of a variable is simple in C++ language. It begins with the data type, followed by at least a space, followed by the name of a variable with a semicolon. Space is optional before the semicolon.



- Variable
  - A memory location whose content may change during program execution
- Declaration:
  - Syntax:

    ```
    type identifier;
    ```
  - Example:

    ```
    double x;
    int y = 45;
    ```

Note optional initialization of the variable

For Example

```
int my_Age;

int my_Salary;

int _employee;
```

If different variables are using the same data type on the same line are also allowed, but care must be taken to separate two with a comma except the last one that would end with a semicolon.

For Example

```
int my_Age,my_Salary,_employee;
```

Declaration in C++ are statements, using this definition, variables may be declared anywhere within the program.

Any legal variable name can be used, it helps to give a sensible name that give you fruitful meaning of what they are being used for. Suppose you are given a short program to add two numbers and display the accumulative value. For this, you should use num1 and num2, for two numbers and a third variable name sum for the result. Always use appropriate names, which can prevent you from extra comments. Remember, C++ programmers tend to prefer short variable names.

We know that a variable has name and type at execution time, it has a location and value. A **constant in C++** means an unchanging value and each constant has a type but does not have location except the string constant.

Numeric Constants

Integer constants consist of one or more digits such as 0,1,2,3,4 or -115. Floating point constants contain a decimal point such as 4.15, -10.05. It can also be written in scientific notation such as 1E-35 means 1*10^-35 or -1E35 means 1*10^35.

Character Constants

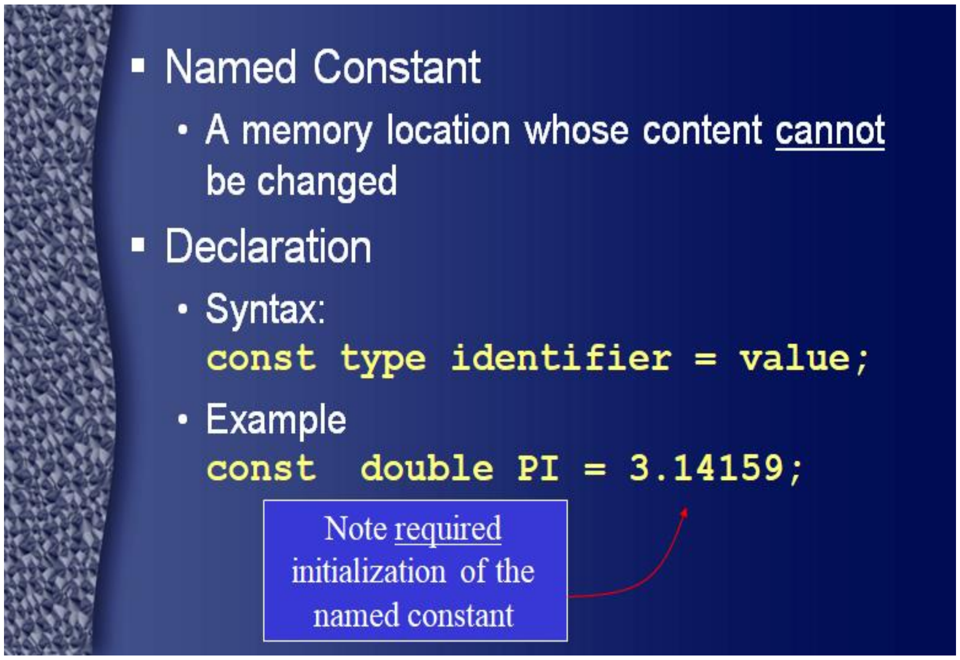
Character constants specify the numeric value of that particular character such as ‘a’ is the value of a.

String Constants

String constants consist of characters enclosed in double quotes such as

“Hello, World”

The string is stored in the memory and the numeric value of that constant is the address of this memory. The string constant is suffixed by ‘\0’, (the null character) by the compiler.



- Named Constant
  - A memory location whose content cannot be changed
- Declaration
  - Syntax:

```
const type identifier = value;
```

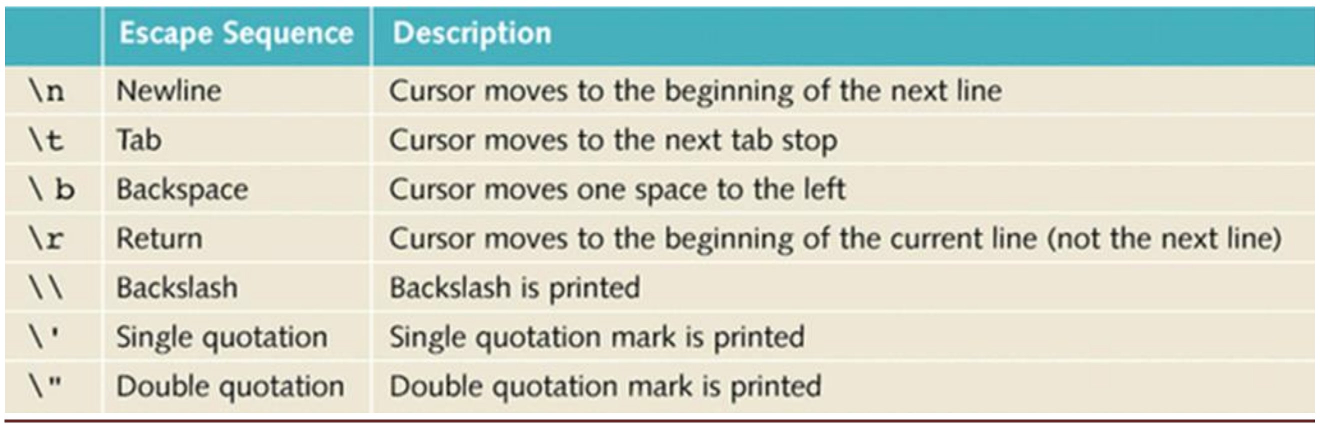
  - Example

```
const  double PI = 3.14159;
```

Note required initialization of the named constant

Both C and C++ use escape sequence in the same manner such as ‘\n’ character to produce a new line. All escape sequences are preceded by a backslash, which indicates a special sequence to the compiler. The compiler as a single character views each and every escape sequence. It seems and may have been expected that an escape sequence occupies 2 bytes which is wrong, it occupies only one byte.

A list of backslash character constant or escape sequence is as in the below table:



	Escape Sequence	Description
\n	Newline	Cursor moves to the beginning of the next line
\t	Tab	Cursor moves to the next tab stop
\ b	Backspace	Cursor moves one space to the left
\r	Return	Cursor moves to the beginning of the current line (not the next line)
\\	Backslash	Backslash is printed
\'	Single quotation	Single quotation mark is printed
\"	Double quotation	Double quotation mark is printed

Defining Constants

There's another way to define constants that dates back to early versions of the C language, the precursor of C++. The preprocessor directive **#define** can create a constant by specifying its name and value, separated by spaces:

```
#define KILLBONUS 5000
```

The constant does not have a type such as int or char. The **#define** directive enables a simple text substitution that replaces every instance of **KILLBONUS** in the code with 5000. The compiler sees only the end result.

Initialization of variables and constants

Most programming languages provide ways of specifying initial values; that is, the values that variables have when program execution begins. Constants must be initialized at the time they are declared, and we have the option of initializing the variables.

Specifying initial values is simple. In its declaration, we follow a variable's name with an equal sign and a constant expression for the desired value. Thus

```
int   temperature = -55;
```

declares Temperature to be an integer, and gives it an initial value of -55.

The compiler initializes static constants simply by defining its value in memory. In the following example, J is a static constant (actually K is a literal)

```
const   int   J=96;   // 32 bit constant

#define K 97
```

H.W

- **Write a program in C++ to explain how to use constants and variables in the simplest way.**
- **Write a program in C++ to explain how to use (#define) to define constants.**

Resources

- **E–books and topics found on scientific and software websites on the Internet**

Southern Technical University

Basra Technical Institute

Department of Computer Networks
and Software Technologies

الجامعة التقنية الجنوبية

المعهد التقني التكنولوجي – البصرة

قسم تقنيات شبكات وبرامجيات الحاسوب

Learning package

Data Types in C++, and They Represent Methods in Memory

For

First year student

2025-2026

By

Sahar Sami Fadhil

Assistant Lectuerer

Week (6)

Pre-Question

- How to classify data in Nature.

Data types in C++, and the represent methods in memory

C++ Data Types

All variables use data-type during declaration to restrict the type of data to be stored. Therefore, we can say that data types are used to tell the variables the type of data it can store.

Whenever a variable is defined in C++, the compiler allocates some memory for that variable based on the data-type with which it is declared. Every data type requires different amount of memory.

Data types in C++ are mainly divided into two types:

1. **Primitive Data Types**: These data types are built-in or predefined data types and can be used directly by the user to declare variables. Example: int, char, float, bool etc.
2. **Abstract or user defined data type**: These data types are defined by user itself. Like, defining a class in C++ or a structure.

Primitive data types available in C++ are:

- **Integer**: Keyword used for integer data types is **int**. Integers typically requires 4 bytes of memory space and ranges from -2147483648 to 2147483647.
- **Character**: Character data type is used for storing characters. Keyword used for character data type is **char**. Characters typically requires 1 byte of memory space and ranges from -128 to 127 or 0 to 255.

- **Boolean**: Boolean data type is used for storing boolean or logical values. A boolean variable can store either *true* or *false*. Keyword used for boolean data type is **bool**.
- **Floating Point**: Floating Point data type is used for storing single precision floating point values or decimal values. Keyword used for floating point data type is **float**. Float variables typically requires 4 byte of memory space.
- **Double Floating Point**: Double Floating Point data type is used for storing double precision floating point values or decimal values. Keyword used for double floating point data type is **double**. Double variables typically require 8 byte of memory space.
- **Void**: Void means without any value. **void** data type represents a valueless entity. **void** data type is used for those functions which do not return a value.
- **Wide Character**: Wide character data type is also a character data type but this data type has size greater than the normal 8-bit data type. Represented by **wchar_t**. It is generally 2 or 4 bytes long.

We can display the size of all the data types by using the size_of() function and passing the keyword of the data type as argument to this function as shown below:

H.W

- **Write a C++ program to print the capacity of the data types mentioned in the lecture topic. Use a function (size_of()) in your program.**

Resources

- **E–books and topics found on scientific and software websites on the Internet**

Southern Technical University

Basra Technical Institute

Department of Computer Networks and Software Technologies

الجامعة التقنية الجنوبية

المعهد التقني التكنولوجي – البصرة

قسم تقنيات شبكات وبرامجيات الحاسوب

Learning package

Expressions Types in C++ Language Arithmetic Expression

For

First year student

2025-2026

By

Sahar Sami Fadhil

Assistant Lectuerer

Week (7)

Pre-Question

- What types of expressions did you encounter during your previous studies in mathematics and logic?
- Do you remember the precedence of mathematical operators?

Expressions types in C++ language

Arithmetic expression

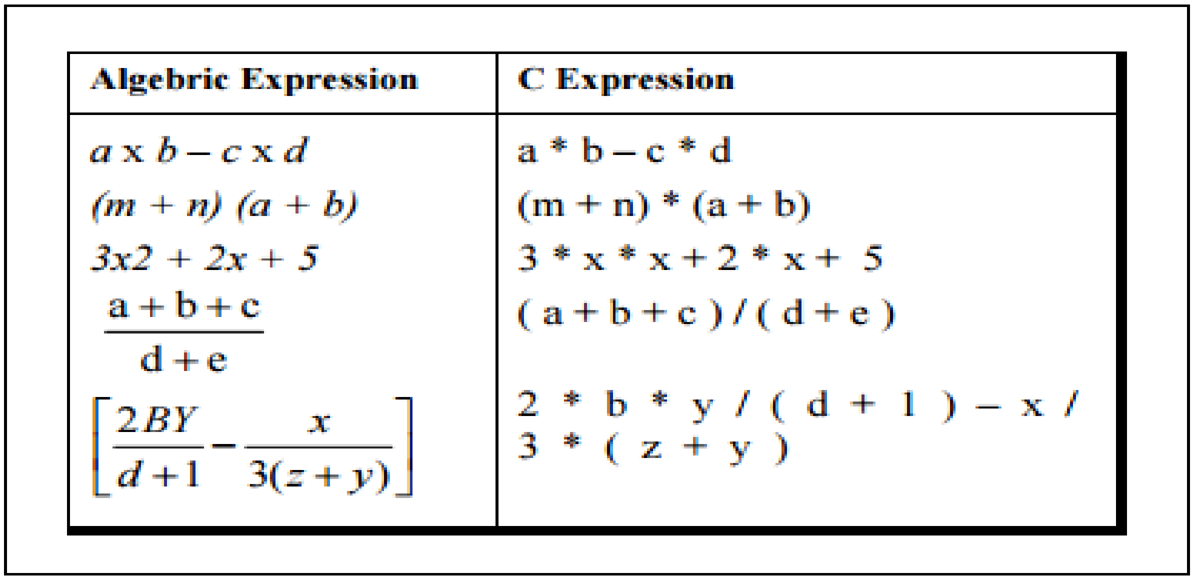
Priority of Arithmetic Operators

The priorities of the operators seen so far are, in high to low priority order:

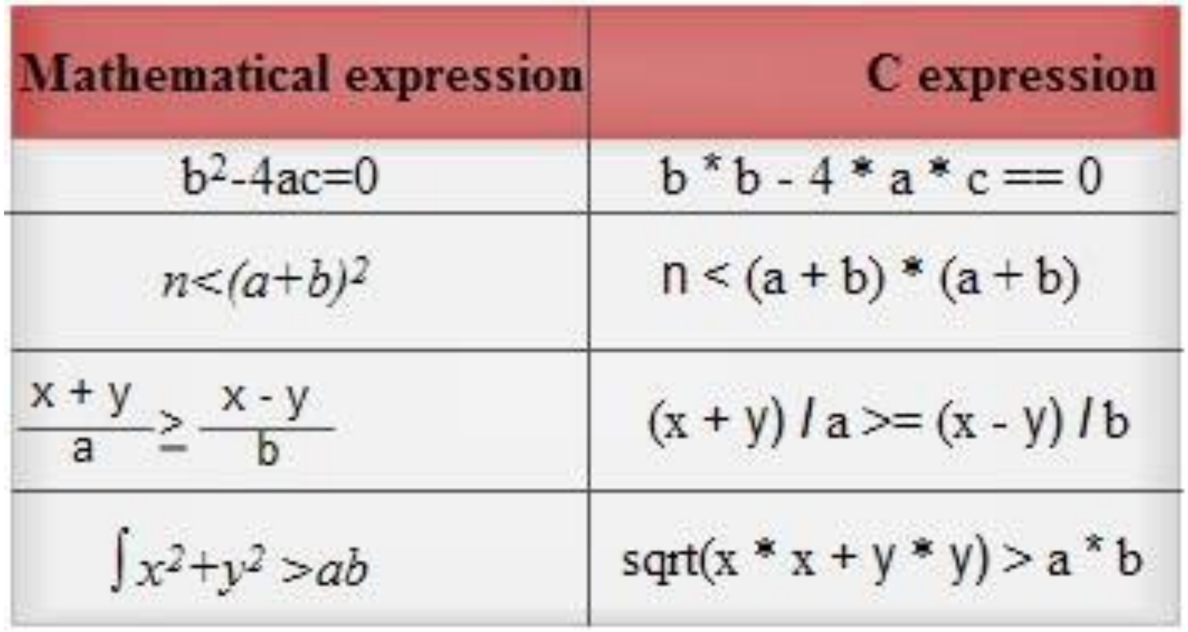
```
 ( )
* / %
+ -
=
```

<u>Conversion manner of Arithmetic Expression to Arithmetic Expression in C++ language</u>

Some Examples



Algebric Expression	C Expression
$a \times b - c \times d$	a * b – c * d
$(m + n)\ (a + b)$	(m + n) * (a + b)
$3x2 + 2x + 5$	3 * x * x + 2 * x + 5
$\frac{a+b+c}{d+e}$	(a + b + c) / (d + e)
$\left[\frac{2BY}{d+1} - \frac{x}{3(z+y)}\right]$	2 * b * y / (d + 1) – x / 3 * (z + y)



Mathematical expression	C expression
b^2-4ac=0	b * b - 4 * a * c == 0
$n<(a+b)^2$	n < (a + b) * (a + b)
$\frac{x+y}{a} \geq \frac{x-y}{b}$	(x + y) / a >= (x - y) / b
$\int x^2+y^2 > ab$	sqrt(x * x + y * y) > a * b

Mathematical Formula	C++ Expression
$b^2 - 4ac$	b*b - 4*a*c
$x(y+z)$	x*(y + z)
$\frac{1}{x^2+x+3}$	1/(x*x + x + 3)
$\frac{a+b}{c-d}$	(a + b)/(c - d)

H.W

- Explain the difference between the sequence of operations performed for the following two mathematical expressions.

$w = x + \frac{y\,a}{z}$ $w = x + \frac{y}{z\,a}$

Resources

- E–books and topics found on scientific and software websites on the Internet

Southern Technical University

Basra Technical Institute

Department of Computer Networks and Software Technologies

الجامعة التقنية الجنوبية

المعهد التقني التكنولوجي – البصرة

قسم تقنيات شبكات وبرامجيات الحاسوب

Learning package

Relational Expression & Logical Expression

For

First year student

2025-2026

By

Sahar Sami Fadhil

Assistant Lectuerer

Weeks (8&9)

Pre-Question

- **Do you remember the relational and logical operators you encountered in your previous studies?**
- **What are the components of a relational expression and a logical expression?**

Relational expression/ relational operations and it's priorities

Relational operators and relational expressions

"A relational operator with constants and variables makes relational expression or An expressions in which relational operators are use is called relational expression."

Points about relational operators

- 1.Relational operators are used to compare values.
- 2.All the expressions evaluates from left to right.
- 3.There are six relational operators in C++ programming (>,< ,>=,<=,==,!=).
- 4.These operators evaluates results true or false.
- 5.False statement represent by 0 and True statement represent by 1.

A simple relational expression contains only one relational operator and takes the following form:

exp1 relational operator exp2

C++ Relational Operators Precedence

<	relational less than
<=	relational less than or equal to
>	relational greater than
>=	relational greater than or equal to
==	relational is equal to
!=	relational is not equal to

The following are examples of relational expressions built from relational operators:

3 < 4	true
7.6 <= 9	true
4 == 7	false
8.3 != 2.1	true

The following two statements print "1" on the standard output

```
bool  b = 8.3 != 2.1;

cout<<"b="<<b<<endl<<endl;
```

A much more common use of relational expressions is to control program flow in if statements or in while, do-while, or for loops.

Of course, relational expressions can also employ variables, as in the following sequence of C++ statements :

```
int x = 3;

int y = 4;

bool b = x > y;

cout<<"b="<<b<<endl<<endl;
```

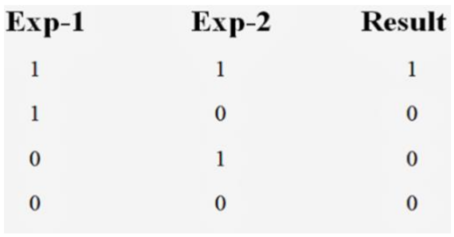
In the third line, the relational expression x > y is evaluated and the result, "0", is assigned to the Boolean variable b. The fourth statement prints "0".

Logical expression/ Logical operations and it's priorities

- 1.There are three logical operators And(&&),or(||) these two both are binary operator and not(!) is unary operator.
- 2. More than one relation expressions are combine by using logical operators.
- 3. The expression will evaluate from left to right if more than one relation expression are use.

And operator (&&)

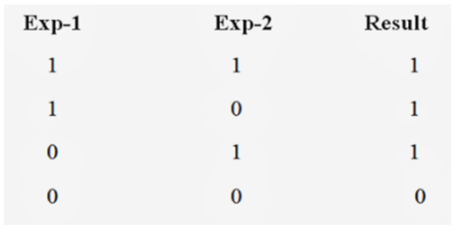
Below table shows evaluation method of and operator, 1 represent True , 0 represent false.



Exp-1	Exp-2	Result
1	1	1
1	0	0
0	1	0
0	0	0

Or operator(||)

Below table shows evaluation method of Or(||) operator.



Exp-1	Exp-2	Result
1	1	1
1	0	1
0	1	1
0	0	0

Not operator(!)

Evaluation method.

Exp	**! Exp (Result)**
1	0
0	1

C++ Logical Operators Precedence

!	unary logical negation
&&	logical AND
||	logical OR

Logical operators are similar to relational operators in that they both produce Boolean results. However, they differ in that logical operators also use Boolean arguments. So the following statements make sense :

Example 1

```
bool   a = true;
bool   b = false;
bool   c = a && b;
cout<<"a && b "<<c<<endl<<endl;
```

Example 2

```
int   a = 3;
int   b = 4;
bool   c = a && b;
cout<<"a && b "<<c<<endl<<endl;
```

Example 3

```
int   a = 3;
int   b = 4;
int   c = 5;
int   d = 6;
bool   e = a < b && c < d;
cout<<"a < b && c < d "<<e<<endl<<endl;
```

We acknowledge the precedence relationship between the relational and logical operators. The relational operators are of higher precedence than the logical && and logical || operators. The parentheses in the following statement illustrate this (although they are not necessary):

```
bool   e = (a < b) && (c < d);
```

The expression (a < b) && (c < d) is true, and this result is assigned to the Boolean variable e .

The following statements print "false" (0) on the standard output.

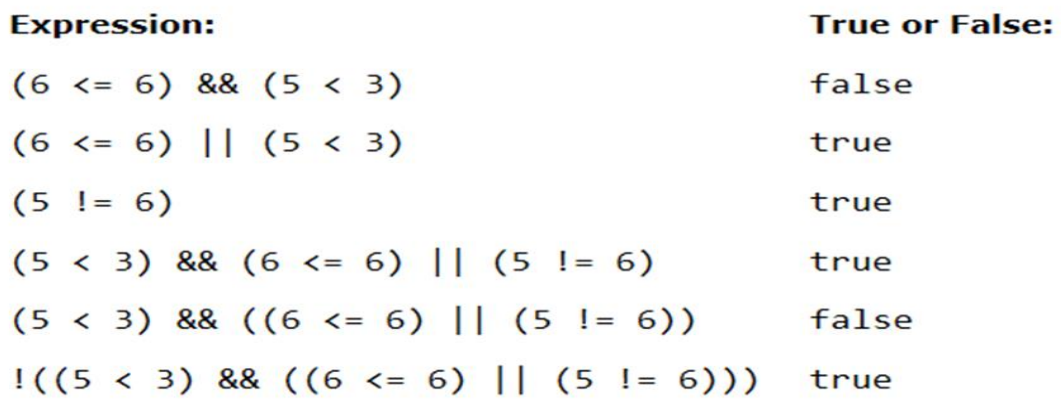
```
bool    b = true;

cout<<"!b ➔"<<!b<<endl<<endl;
```

Note the ! operator does not change the value of the Boolean variable b; it simply returns a value which is the logical complement of b. It is of higher precedence than all the logical or relational operators.

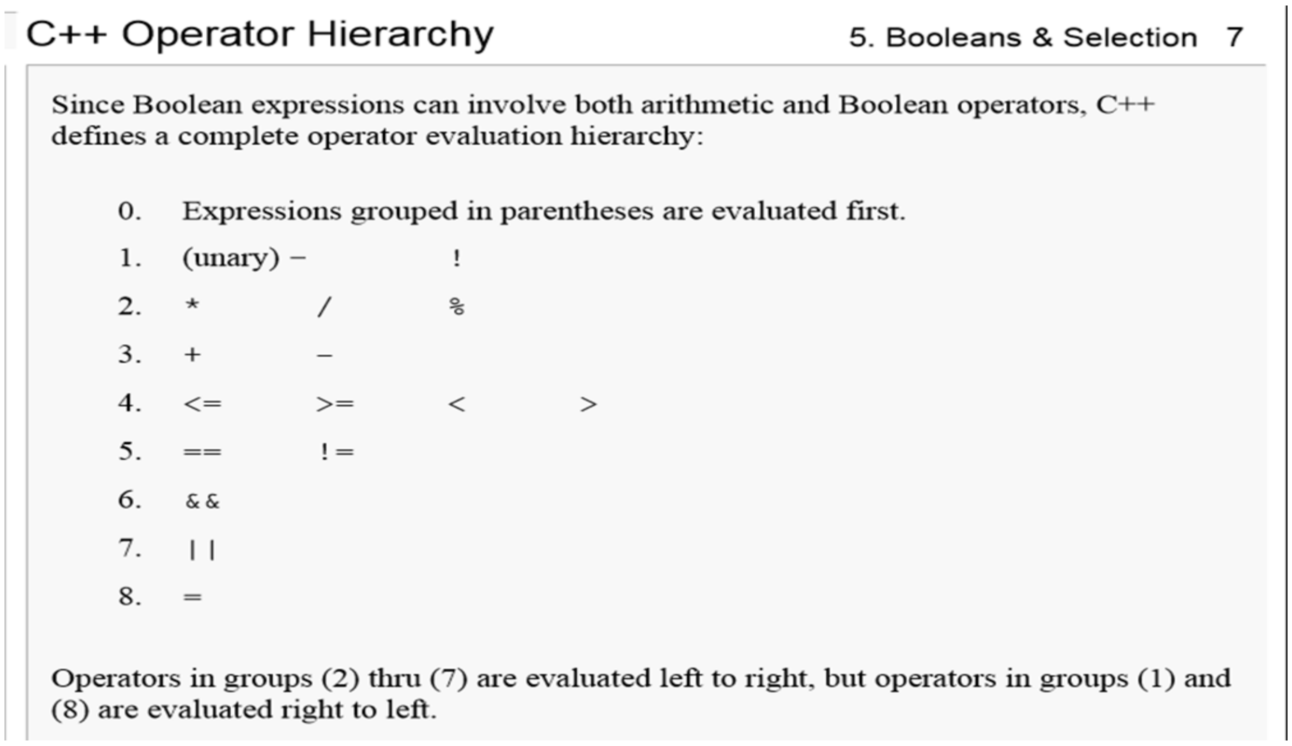
Compound Expressions/ Priorities table of public operations/ deferent examples

Expressions can be combined using the logical operators "&&" ("and"), "||" ("or") and "!" ("not"), as in the following examples :



Expression:	True or False:
(6 <= 6) && (5 < 3)	false
(6 <= 6) || (5 < 3)	true
(5 != 6)	true
(5 < 3) && (6 <= 6) || (5 != 6)	true
(5 < 3) && ((6 <= 6) || (5 != 6))	false
!((5 < 3) && ((6 <= 6) || (5 != 6)))	true

The fourth of these expressions is true because the operator "&&" has a higher precedence than the operator "||".



C++ Operator Hierarchy   5. Booleans & Selection   7

Since Boolean expressions can involve both arithmetic and Boolean operators, C++ defines a complete operator evaluation hierarchy:

0.	Expressions grouped in parentheses are evaluated first.			
1.	(unary) –		!	
2.	*	/	%	
3.	+	–		
4.	<=	>=	<	>
5.	==	!=		
6.	&&			
7.	||			
8.	=			

Operators in groups (2) thru (7) are evaluated left to right, but operators in groups (1) and (8) are evaluated right to left.

H.W

- **What is the difference between a relational expression and a logical expression in terms of the inputs and outputs of each expression?**

Resources

- **E-books and topics found on scientific and software websites on the Internet**

Southern Technical University

Basra Technical Institute

Department of Computer Networks and Software Technologies

الجامعة التقنية الجنوبية

المعهد التقني التكنولوجي – البصرة

قسم تقنيات شبكات وبرامجيات الحاسوب

Learning package

Spaces, Brackets, Type of Comments Increment, Decrement Operators

For

First year student

2025-2026

By

Sahar Sami Fadhil

Assistant Lectuerer

Week (10)

Pre-Question

- **What are non-executive statements?**

C++ Comments

Each and every language will provide this feature which is used to document source code. **Comment is non-executable Statement in the C++**.

Types of comments

There are two types of comments used in C++

1. Single line comments:

Single line comments is accomplished by double-slash (//). Everything that is followed by double-slash till the end of line is ignored by the compiler.

2. Multi-line comments:

Multi-line comments starts by using forward slash followed by asterisk (/*) and ends by using asterisk followed by forward slash (*/). Everything between (/*) and (*/) are ignored by compiler whether it is one or more than one line.

Examples and Notes

```
cout<<"Hello"; //Print Hello Word

cout<<"www.c4learn.com"; //Website

cout<<"Pritesh Taral"; //Author




main()

{

/* this comment

        can be considered as

   multiple line comment */

cout << "Hello C++ Programming";

}
```

Increment/decrement Operators

Increment operators are used to increase the value of the variable by one and decrement operators are used to decrease the value of the variable by one in C programs.

Increment operator: + + i ; i + + ;

Decrement operator: – – i ; i – – ;

H.W

- How can we use comments in C++ to track the execution of long programs?
- What is the difference between the pre- and post-use of the ++ or -- operator?

Resources

- E–books and topics found on scientific and software websites on the Internet

Southern Technical University

Basra Technical Institute

Department of Computer Networks and Software Technologies

الجامعة التقنية الجنوبية

المعهد التقني التكنولوجي – البصرة

قسم تقنيات شبكات وبرامجيات الحاسوب

Learning package

Assignment Statement Types

For

First year student

2025-2026

By

Sahar Sami Fadhil

Assistant Lectuerer

Week (11)

Pre-Question

- **What is an assignment statement? Have you ever encountered one while studying mathematics?**

Special Assignment Expressions

An expression can be categorized further depending upon the way the value are assigned to the variables

1-Chained assignment: Chained assignment is an assignment expression in which the same value is assigned to more than one variable, using a single statement.

```
a = (b=20);   or     a=b=20;
```

Note that variables cannot be initialized at the time of declaration using chained assignment.

```
int a=b=30; // illegal
```

2-Embedded assignment: Embedded assignment is an assignment expression, which is enclosed within another assignment expression.

a=20+(b=30);   //equivalent to   b=30; a=20+30 ;

Note that the expression (b=30) is an embedded assignment.

3-Compound Assignment: Compound Assignment is an assignment expression, which uses a compound assignment operator that is a combination of the assignment operator with a binary arithmetic operator.

a + =20; //equivalent to a=a+20 ;

In this statement, the operator += is a compound assignment operator, also known as **short-hand** assignment operator.

Short-Hand Assignment operators

C++ has a set of shorthand assignment operators of the form.

var oper= exp;

Here var is a variable, exp is an expression and oper is a C++ binary arithmetic operator. The operator **oper =** is known as short-hand assignment operator.

Example :

x += 1 is same as x = x + 1

The commonly used shorthand assignment operators are as follows:

x += y → x = x + y   x -= y → x = x – y   x *= y → x = x * y

H.W

- Write a program in C++ using the types of assignment statements you learned in lecture.

Resources

- E–books and topics found on scientific and software websites on the Internet

Southern Technical University

Basra Technical Institute

Department of Computer Networks and Software Technologies

الجامعة التقنية الجنوبية

المعهد التقني التكنولوجي – البصرة

قسم تقنيات شبكات وبرامجيات الحاسوب

Learning package

Comma Operator & Conditional Operator

For

First year student

2025-2026

By

Sahar Sami Fadhil

Assistant Lectuerer

Week (12)

Pre-Question

- Can you write more than one expression or assignment statement on a single line in your program?
- Do all programs execute in the same way, or is there a possibility that the program will take one of two different paths depending on the value of a particular condition?

The Comma Operator

You can use the comma operator to include several expressions where a single expression is syntactically correct. The following syntax applies for the comma operator:

Syntax: expression1, expression2 [, expression3...]

The expressions separated by commas are evaluated from left to right.

Example:

```
int  x, i, limit;

x = i * i,  cout << x << endl << endl ;
```

The comma operator separates the variables x and i and limit and is then used to calculate and output the value of x in a single statement. The comma operator has the lowest precedence of all operators — even lower than the assignment operators.

Example:

x = (a = 3, b = 5, a * b);

In this example the statements in brackets are executed before the value of the product of a * b is assigned to x.

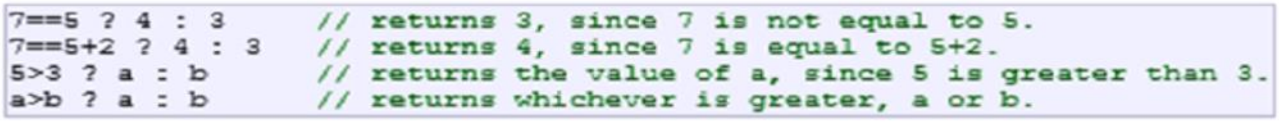
Conditional operator (?)

The conditional operator evaluates an expression returning a value if that expression is true and a different one if the expression is evaluated as false. Its format is :

condition ? result1 : result2

If condition is true the expression will return result1, if it is not it will return result2.

Examples



```
7==5 ? 4 : 3      // returns 3, since 7 is not equal to 5.
7==5+2 ? 4 : 3    // returns 4, since 7 is equal to 5+2.
5>3 ? a : b       // returns the value of a, since 5 is greater than 3.
a>b ? a : b       // returns whichever is greater, a or b.
```

```
int  a, b, c;
a=2;
b=7
c = ( a > b ) ? a : b ;
cout << “ c = “ << c ;
```

In this example the value of c is 7.

H.W

- What is the difference between the compiler and an interpreter?
- What is the classification of the C++ language within the programming language levels?

Resources

- E–books and topics found on scientific and software websites on the Internet

Southern Technical University

Basra Technical Institute

Department of Computer Networks and Software Technologies

الجامعة التقنية الجنوبية

المعهد التقني التكنولوجي – البصرة

قسم تقنيات شبكات وبرامجيات الحاسوب

Learning package

Conditional Statement

if, if-else, Nested if-else conditional statement

For

First year student

2025-2026

By

Sahar Sami Fadhil

Assistant Lectuerer

Weeks (13&14)

Pre-Question

- **Can you use the condition operator (?:) when you have more than one statement to execute if the answer to the condition is yes or no?**

Control, conditional and loop statements (Control Structures)

A program is usually not limited to a linear sequence of instructions. During its process it may bifurcate, repeat code or take decisions. For that purpose, C++ provides control structures that serve to specify what has to be done by our program, when and under which circumstances .

With the introduction of control structures we are going to have to introduce a new concept: the compound statement or block. A block is a group of statements which are separated by semicolons (;) like all C++ statements, but grouped together in a block enclosed in braces :{ } :

```
{
  statement1;
  statement2;
  statement3;
}
```

Most of the control structures require a generic statement as part of its syntax. A statement can be either a simple statement (a simple instruction ending with a semicolon) or a compound statement (several instructions grouped in a block), like the one just described. In the case that we want the statement to be a simple statement, we do not need to enclose it in braces ({}). But in the case that we want the statement to be a compound statement it must be enclosed between braces ({}), forming a block.

Conditional Statements

if statement and if … else statement

The **if** keyword is used to execute a statement or block only if a condition is fulfilled. Its form is :

Where condition is the expression that is being evaluated. If this condition is true, statement is executed. If it is false, statement is ignored (not executed) and the program continues right after this conditional structure. For example, the following code fragment prints x is 100 only if the value stored in the x variable is indeed 100 :

```
if (x == 100)   cout << "x is 100"<<endl<<end;
```

If we want more than a single statement to be executed in case that the condition is true we can specify a block using braces :{ }

```
if (x == 100)
   {
        cout << "x is ";
        cout << x  ;
   }
```

We can additionally specify what we want to happen if the condition is not fulfilled by using the keyword else. Its form is :

```
if (condition) statement1 else statement2
```

For example :

```
if (x == 100)
    cout << "x is 100";
else
    cout << "x is not 100 " ;
```

prints on the screen x is 100 if indeed x has a value of 100, but if it has not -and only if not- it prints out x is not 100 .

The (if … else) structures can be concatenated with the intention of verifying a range of values. The following example shows its use telling if the value currently stored in x is positive, negative or none of them (i.e. zero) :

```
if (x > 0)
    cout << "x is positive";
else if (x < 0)
    cout << "x is negative";
else
    cout << "x is 0 " ;
```

Remember that in case that we want more than a single statement to be executed, we must group them in a block by enclosing them in braces { }.

H.W

- Rewrite all programs in which you used the condition operator (?:) and replace it with the if statement.
- Write a C++ program that demonstrates the use of nested if statements.

Resources

- E–books and topics found on scientific and software websites on the Internet

Southern Technical University

Basra Technical Institute

Department of Computer Networks and Software Technologies

الجامعة التقنية الجنوبية

المعهد التقني التكنولوجي – البصرة

قسم تقنيات شبكات وبرامجيات الحاسوب

Learning package

Switch Conditional statement

For

First year student

2025-2026

By

Sahar Sami Fadhil

Assistant Lectuerer

Week (15)

Pre-Question

- Have you ever faced any difficulty when writing a program that contains nested if-else statements?

The selective structure- switch

The syntax of the switch statement is a bit peculiar. Its objective is to check several possible constant values for an expression. Something similar to what we did with the concatenation of several if and else if instructions. Its form is the following:

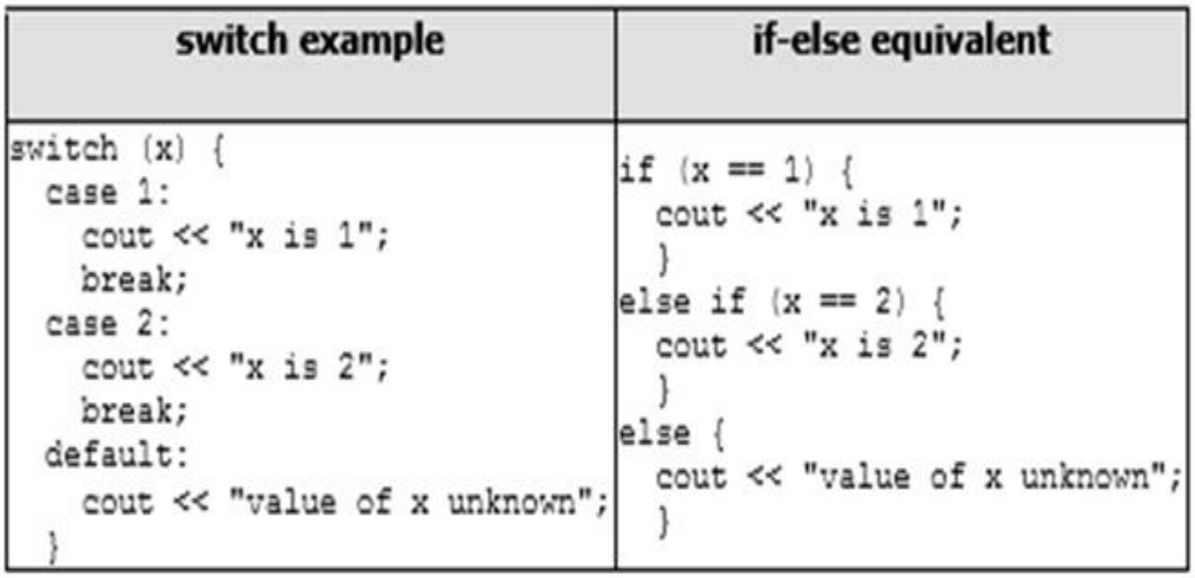
```
switch (expression)
{
  case constant1:
     group of statements 1;
     break;
  case constant2:
     group of statements 2;
     break;
  .
  .
  .
  default:
     default group of statements
}
```

It works in the following way: **switch** evaluates expression and checks if it is equivalent to constant1, if it is, it executes group of statements 1 until it finds the **break** statement. When it finds this **break** statement the program jumps to the end of the switch selective structure .

If expression was not equal to constant1 it will be checked against constant2. If it is equal to this, it will execute group of statements 2 until a break keyword is found, and then will jump to the end of the switch selective structure .

Finally, if the value of expression did not match any of the previously specified constants (you can include as many case labels as values you want to check), the program will execute the statements included after the **default:** label, if it exists (since it is optional).

Both of the following code fragments have the same behavior:

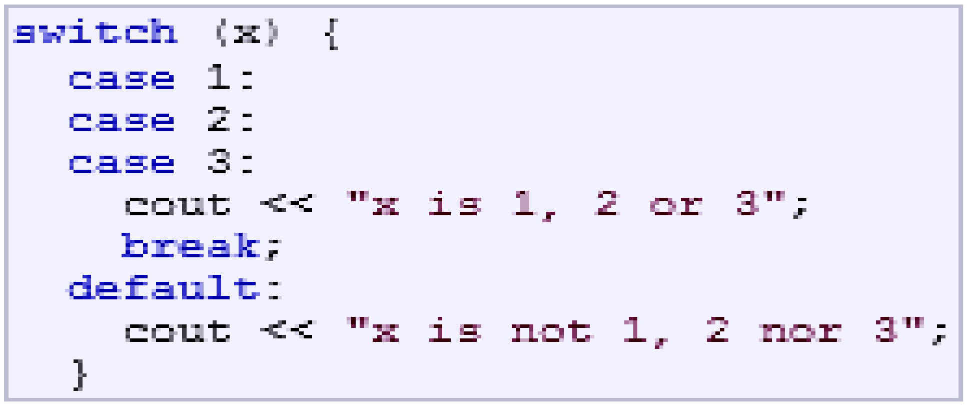


switch example	if-else equivalent

`switch (x) {`
`  case 1:`
`    cout << "x is 1";`
`    break;`
`  case 2:`
`    cout << "x is 2";`
`    break;`
`  default:`
`    cout << "value of x unknown";`
`  }`
	`if (x == 1) {`
	`  cout << "x is 1";`
	`  }`
	`else if (x == 2) {`
	`  cout << "x is 2";`
	`  }`
	`else {`
	`  cout << "value of x unknown";`
	`  }`

The switch statement is a bit peculiar within the C++ language because it uses labels instead of blocks. This forces us to put break statements after the group of statements that we want to be executed for a specific condition. Otherwise the remainder statements -including those corresponding to other labels- will also be executed until the end of the switch selective block or a break statement is reached.

For example, if we did not include a break statement after the first group for case one, the program will not automatically jump to the end of the switch selective block and it would continue executing the rest of statements until it reaches either a break instruction or the end of the switch selective block. This makes unnecessary to include braces { } surrounding the statements for each of the cases, and it can also be useful to execute the same block of instructions for different possible values for the expression being evaluated. For example:



```
switch (x) {
  case 1:
  case 2:
  case 3:
    cout << "x is 1, 2 or 3";
    break;
  default:
    cout << "x is not 1, 2 nor 3";
  }
```

Notice that switch can only be used to compare an expression against constants. Therefore we cannot put variables as labels (for example case n: where n is a variable) or ranges (case (1..3):) because they are not valid C++ constants .

If you need to check ranges or values that are not constants, use a concatenation of if and else if statements.

H.W

- **Can all programs written with nested if statements be converted to programs written with a switch statement?**
- **Choose a previous program written with nested if statements that can be rewritten with a switch statement, write it and then execute it.**

Resources

- **E–books and topics found on scientific and software websites on the Internet**