

CHAPTER

C 1

إعداد : محمد مهيدات
اسامة الوضني



مقارنة بين اللغة العادية ولغة البرمجة

1.1 Natural language vs. programming language

- What is a language?
 - a language is a tool for expressing and recording human thoughts
- What is a programming language?
 - A programming language is defined by a certain set of rigid rules, these rules determine which symbols (letters, digits, punctuation marks, and so on) could be used in the language

تعريف اللغة بشكل عام : هي أداة لوصف وتدوين أفكار البشر بينما لغة البرمجة تعرف بأنها مجموعة من القواعد „ هاي القواعد بتتعدد الرموز (الأحرف , الأرقام , علامات الترقيم) يلي ممكن نستخدمها في هاي اللغة.

1.1 Natural language vs. programming language

- Any programming language consists of:
 - **Syntax:** is a set of rules determines the appropriate ways of collating the symbols
 - **Semantic:** the ability to recognize the meaning of every statement expressed in the given language

Any program we write must be error-free in these three ways: lexically, syntactically and semantically, otherwise, the program won't run, or it will produce unacceptable results

أي لغة برمجة تتكون من :-

1 - **القواعد (syntax)** : وهي مجموعة من القواعد يلي بتتعدد الطريقة يلي من خلالها بتبيّن رموز هاللغة. مثل قواعد اللغة العربية ما تقدر تنصب الفاعل مثل

2 - **دللات الألفاظ :** هي القدرة انك تدرك معنى كل جملة موصوفة بلغة معينة. يعني يكون الجملة لها معنى ، يعني بالعربي م بقدر ادكي شربت السفينة التفاحه!

أي برنامج بنكتبه لازم يكون خالي من الاخطاء : ك مفردات و ك قواعد و ك دلالات لفظية ، غير هيك البرنامج م رح يستغل ، او اذا اشتغل رح يعطي نتائج غير مقبولة

1.1 Natural language vs. programming language

- Why we need to use a programming language?
 - The machine language is the simplest and the most primary language we can use to give commands to our computer (very **difficult to understand for humans**)
 - A **high-level programming language** is a bridge between the people's language (natural language) and computer language (machine language)
 - A **high-level programming language** is similar to a natural language: it uses symbols, words and conventions readable to humans.
 - High-level languages include Basic, FORTRAN, COBOL, Pascal, C, C++, C#, and Java

لِيُشْ بَنْدَتَاج لُغَةُ الْبَرْمَجَةِ؟!

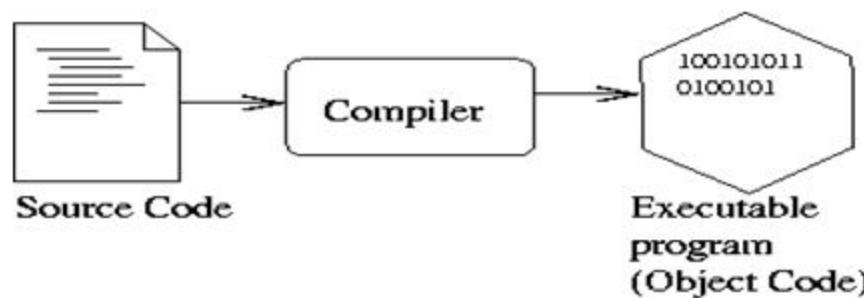
لغة الآلة هي الأبسط (تستخدم 01 فقط) وهي أكثر لغة بدائية، وهي بنقد من خلالها نعطي اوامر للكمبيوتر، طبعاً صعبة جداً انه البشر يفهموها، يعني لو دكتيرك 01001010 كيف بده تستوعب على !!

لغات البرمجة المتقدمة (يلي بقدروا يفهموها البشر) : هي جسر بين لغة الناس ولغة الكمبيوتر وهي شبيهة جداً للغة الطبيعية ، فهي بتستخدم رموز وكلمات قابلة للقراءة عند البشر
أمثلة على لغات برمجة متقدمة :

Basic, FORTRAN, COBOL, Pascal, C, C++, C#, and Java

1.1 Natural language vs. programming language

- Compiler: a specialized computer program translates a program written in a high-level language to machine language



الكومبایلر : هو برنامج حاسوبي بدول البرنامج من اللغة المتقدمة للغة الآلة يعني لما تكتب برنامج مثلاً بلغة آل ++ ، الكومبایلر رح يدول الكود تاعك من لغة الـ ++ للغة الآلة ، مشان يقدر يفهمها الكمبيوتر .

Processing a C++ Program

```
#include <iostream>
using namespace std;
int main()
{
    cout << "My first C++ program." << endl;
    return 0;
}
```

Sample Run:
My first C++ program.

هذا برنامج بسيط يذلل الكمبيوتر يطبع على الشاشة

(My first C++ program.)

Processing a C++ Program (cont'd.)

- To execute a C++ program:
 - Use an editor to create a source program in C++
 - Preprocessor directives begin with # and are processed by the preprocessor. (A set of preliminary information that the compiler needs is included in **header files** such **iostream** header file)
 - Use the compiler to:
 - Check that the program obeys the rules
 - Translate into machine language (object program)
 - Linker:
 - Combines object program with other programs provided by the SDK to create executable code
 - Loader:
 - Loads executable program into main memory
- The last step is to execute the program

مراحل عمل برنامج C++ بتحتاج إلى :

repl.it : وهو البرمجية اللي بتسمح لك تكتب كود C++ مثل visual studio او مواقع الالونلين مثل repl.it

-2 Preprocessor توجيهات المعالج

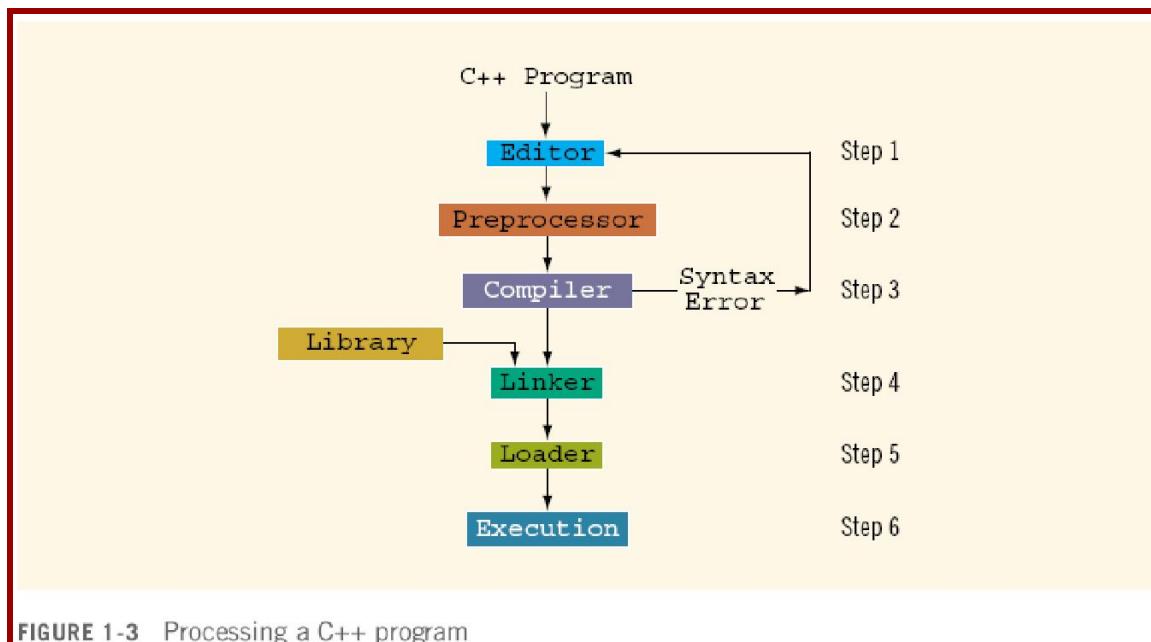
هاري التوجيهات بتبدأ برمز (#) بمعالجها المعالج ومنها **header files** (ملفات بحتاجها المعالج) مثل **iostream** وهي اختصار لـ Input/Output Stream ، وهي المكتبة الخاصة لدخول المعلومات من المستخدم وارجاع الطباعة (لازم تكون موجودة إذا بدخل بيانات من المستخدم وإذا بدهك تطبع على الشاشة)

-3 compiler

بنستخدمه لشغليين : الاولى إننا نتأكد انه الكود بخضع لقواعد اللغة ، والثانية مثل م دكينا يدول الكود للغة الآلة .

-4 : هو بربط البرنامج اللي دوله الكومبایلر للغة الآلة ببرامج اخرى موجودة باشي اسمه SDK وهي مجموعة مكتبات بتساعد بتنفيذ الكود (مش مهم تعرفها)

-5 : وهو اللي يرفع البرنامج الجاهز للتنفيذ للرام وآخر خطوة هي تنفيذ البرنامج ..



هاد الشكل بوضحتنا خطوات تنفيذ البرنامج ، لكن في شكل أشمل منه رح نشوفه كمان شوي

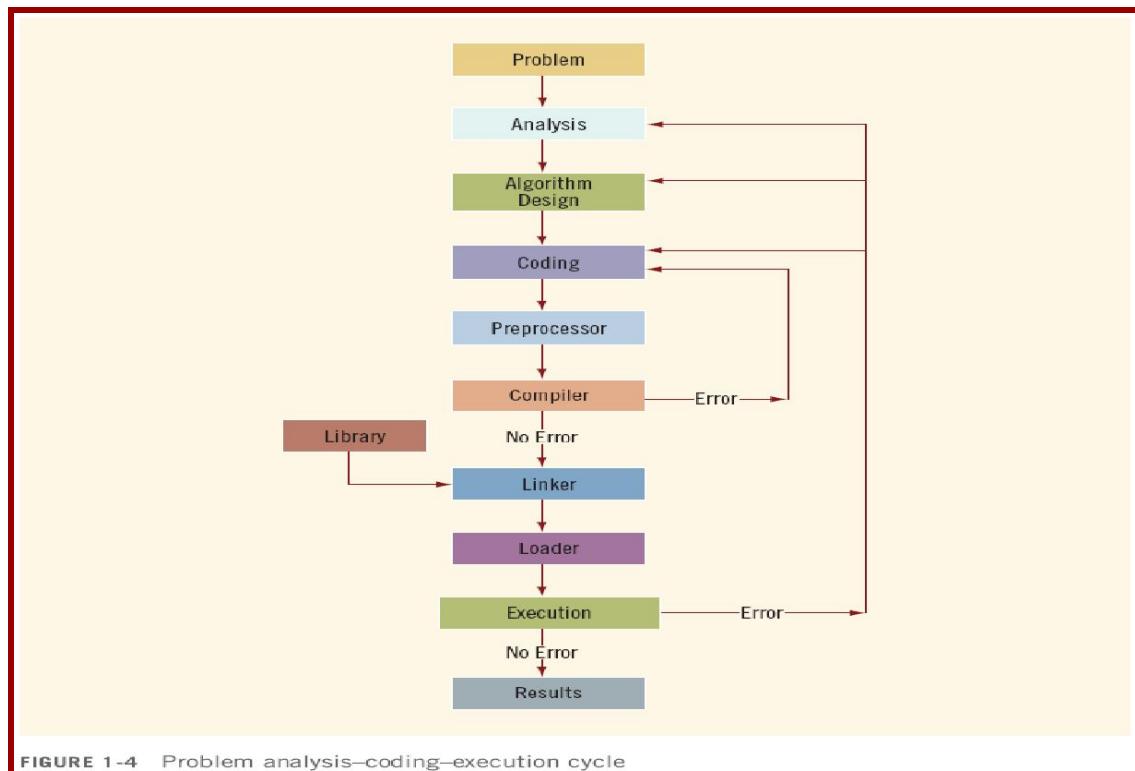
(ملاحظة : الدكي الي ملون بالاصل يعني **مش مهم** كثير)

"Programming is a process of problem solving
One problem-solving technique:
Analyze the problem
Outline the problem requirements
Design steps (algorithm) to solve the problem
Algorithm:
Step-by-step problem-solving process
Solution achieved in finite amount of time"

البرمجة هي عبارة عن عملية لحل المشاكل
(مش مثل ما بظن الأغلب ، إنها عملية كتابة كود ، كتابة الكود خطوة فقط من البرمجة)

خطوات حل المشكلة :

- 1 - تحليل المشكلة
 - 2 - بتندرج المطلوب منها
 - 3 - بتبعد خوارزمية للحل
- والخوارزمية : هي خطوات لحل المشكلة والوصول للحل بوقت محدد.



- Run code through compiler
- If compiler generates errors
 - Look at code and remove errors
 - Run code again through compiler
- If there are no syntax errors
 - Compiler generates equivalent machine code
 - Linker links machine code with system resources
- Once compiled and linked, loader can place program into main memory for execution
- The final step is to execute the program
- Compiler guarantees that the program follows the rules of the language
 - Does not guarantee that the program will run correctly

The Problem Analysis-Coding-Execution Cycle

وهو مهم جدا وبوضاحك خطوات حل المشكلة ، يلي هي التحليل للمشكلة ، ومن ثم وضع خوارزمية للحل ومن ثم كتابة هاي الخوارزمية بلغة الـ C++

، وبرضو بوضوح خطوات تنفيذ البرنامج يلي ذكرناهم قبل ، لكن في اشي جدير بالذكر ، انه لو كان **عندی خطأ بالقواعد** . رح يكون الخطأ اكيد بكتابته الكود يعني بمراحله ال coding والبرنامج ما بشتغل بالمرة . **Syntax Error**

بينها لو حصلت على نتائج خطأ **بعد تنفيذ البرنامج** **Logical Error** رح يكون سبب الخطأ الله 3 احتمالات:
إما فهم خاطئ للمشكلة أو بعملية تحليلي للمشكلة أو بعملية كتابة الكود

يعني كمثال لو كان مطلوب مني برنامج يحسب مساحة دائرة ، وكتبت السنتاكس بطريقة صريحة ودخلت المدخلات إنه نصف القطر = 2 ، ف لازم الجواب يكون $\pi * 2^2$ ، ف مثلا لو طبع الجواب 15 ، اكيد رح يكون في غلط إما بقراءتي للسؤال (تحليل المشكلة) او بكون مو حافظ القانون لما اجيته اكتب الكود (خوارزمية الحل والcoding)

```
#include <iostream>
using namespace std;
int main(void)
{
    cout << "It's me, your first program.";
    return 0;
}
```

هاد عباره عن برنامج بسيط ، رح نشرح من خلاله بعض الامور :

الجملة يلي مؤشر عليها **السطر** هي مثال على
ال preprocessor directives يلي وضمنها بالصفحات قبل ..

بالنسبة للعبارة يلي بالمستطيل ” ال **namespace** ” هي عباره عن
container (حاوية) بتحتوي على أشياء انا معرفها للبرنامجه يلي
بكتب فيه

، مثلا عندي عندي جامعة اسمها JUST وهالجامعة بتحتوي عدة كليات (IT العز والعلوم الخ...) .. ف أنا هذول بحطهم ب namespace (مش مهم كيف س المهم توصلك الفكرة) وبس آجي استخدمهم ببرنامجه إذا حكيت IT رح يعرف الكومبايلر إنه قصدي عن ال IT يلي ب just ، لأنني عملت just namespace اسمه just وعرفت جواه هاي المعلومات وببداية البرنامج كتبت just; ف الكومبايلر صار متعرف على أي اشي جوا هال namespace just;

ولغة ال C++ ' في الها أدوات خاصة فيها ، مثل الجامعة بالمثال السابق ما عندها كليات خاصة فيها ف هي دخلت كل هالأدوات ب namespace اسمه std .
ف أنا لما أكتب just; الكومبايلر دغري رح يتعرف على كل ال tools الخاصة بلغة ال C++ ، مثل أداة ال cout ، cout يلي رح نشوفهم لقدام ..

بالنسبة لـ **int main (void)** ، هو عبارة عن main function ، يفترض أنه أي برنامج ++c يحتوي عليه ، ولازم يبدأ برمز { وينتهي برمز }

*functions كامل خاص بالـ

نزل للسطر يلي بعده .. ال **cout** هو عبارة عن أداة معرفها بال namespace std ، **بطبع أي شي بعدها على الشاشة** ، وهون اللي كان نص **(string)** واي مكتوب لازم يبدأ وينتهي بلغة ++c بهاد الرمز ("") ، وب Russo لازم أي جملة في لغة الـ ++c تنتهي ب الفاصلة المنقوطة (;

نزل للـ **return 0** ، كلمة return تتدكي لي compiler إنه هون خلص تنفيذ الـ main function ، وأي أمر جوا الـ main مكتوب بعد الـ return رح يتتجاهله الـ compiler ومبنفذت ، بالنسبة لـ 0 ، ف هي اللي اشي يتعلق بالـ int مكتوبة قبل الكلمة main ، ورح نشرحهم بالتفصيل في تشابير الـ functions .

المخرجات رح تكون على الشاشة it's me, your first program

```
main.cpp saved Error saving file
1 #include <iostream>
2 using namespace std;
3 int main() {
4     cout<<"it's me, your first program";
5     return 0;
6 }
```

```
clang version 7.0.0-3~ubuntu0.18.04.1
clang++-7 -pthread -o main main.cpp
./main
it's me, your first program
```

Data Types, variables and Operators

Comments

Comments are for the reader, not the compiler

- To explain to other readers of the code how the tricks used in the code work
- To explain the means of variables and functions
- To document who the author is and when the program was written
- Whenever the compiler encounters a comment in your program, the compiler will skip it to the end of the comment.

Two types:

Single line

```
// This is a C++ program. It prints the sentence:  
// Welcome to C++ Programming.
```

Multiple line

```
/*  
 You can include comments that can  
 occupy several lines.  
 */
```

الكومنتات هي للقارئ ومش للكومبيالر (مش للتنفيذ) ، وممكن نستخدمها لعدة أشياء

- نشرح للي رح يقرأ الكود من بعدهنا كيف صنعنا هالكود
- نبين سبب استخدام كل فاريبل وكل فنكشن (رح نشرحهم الاثنين)
- نوثق مين كتب هالكود وتاريخ كتابة الكود

وأول ما يلاقي الكومبيالر اشارة إنه هون في كومنت ، بيعمل skip عن كل الكومنت وبكميل قراءة اللسدر يلي بعده.

في نوعين للكومنتات ، أول اشي يلي بدأوب // وهاد الكومنت بنتهي بنهاية السطر ، والثاني يلي بدأ ب /* وينتهي بس ينكتب هالرمز */

- Reserved Words (Keywords)
- Reserved words, keywords, or word symbols

- **Include:**

- int
- float
- double
- char
- const
- void
- return

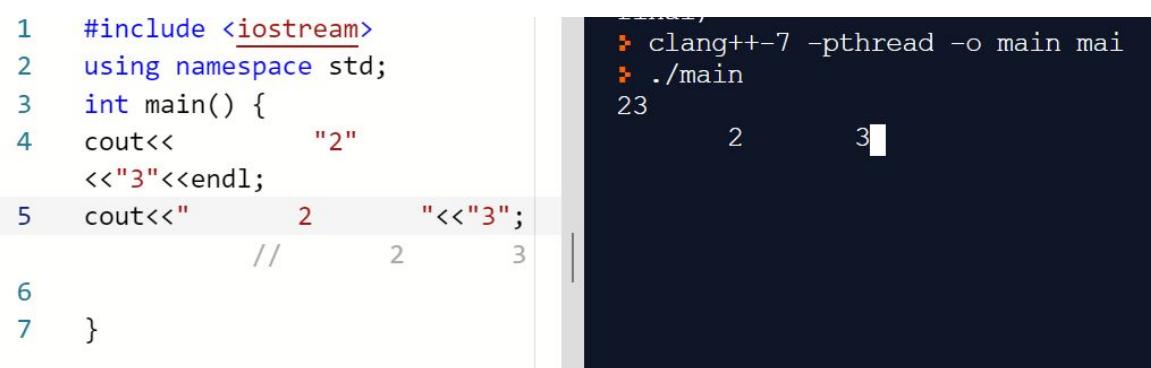
الكلمات المحفوظة (reserved words) هي كلمات خاصة بلغة C++ وهي  بعض الأمثلة محفوظة يعني ما يشير تعرف متغير أو اقتراح باسمها *

▪ Whitespace

- Every C++ program contains whitespaces
 - Include blanks, tabs, and newline characters
- Used to separate special symbols, reserved words, and identifiers
- Proper utilization of whitespaces is important
 - Can be used to make the program readable

الفراغات هي الفراغات بالكود ، بنستخدمها منشأ نفصل الرموز والكلمات المحفوظة عن بعضها البعض ، وبنعمل الكود قابل للقراءة بشكل أكبر.

مثال :



```

1 #include <iostream>
2 using namespace std;
3 int main() {
4     cout<<      "2"
5     <<"3"<<endl;
6     cout<<"      2      " <<"3";
7 }
```

clang++-7 -pthread -o main main.cpp
./main
2
3
2
3

" شوف المرة الاولى طبع 23 جنب بعض ، لأنه م بشوف الفراغات اصلا ، بينما المرة الثانية الفراغ كان جوا ال " ف اعتبره جزء من النص يلي رح يطبعه ."

1.3 Variables

- **Variables:** are special “containers” used to store the results of C++ operations in order to use them in other operations
- As the name *variables* suggests, the content of a container can be varied



المتغيرات: هي (حاويات صغيرة) وظيفتها تخزن قيم ناتجة من عمليات في لغة الـ C++ مشان تستخدموها بعمليات أخرى ، وكوونه اسمها متغيرات ف هالاشي يعني انه القيمة داخل هالمتغير قابلة للتغيير .

1.3 Variables (identifiers)

- Consist of letters, digits, and the underscore character (_)
- Must begin with a letter or underscore
- C++ is case sensitive
 - NUMBER is not the same as number
- Two predefined identifiers are `cout` and `cin`
- Unlike reserved words, predefined identifiers may be redefined, but it is not a good idea

شروط تسمية المتغيرات :-

★ الأشياء اللي مسموح تكون موجودة باسم المتغير

- الحروف (انجليزية طبعا)
- الأرقام
- الرمز (_ -Underscore-)

★ لازم تبدأ بحرف او _ (ممنوع رقم)

★ ممنوع يكون متغيرين بنفس الاسم .

- لغة الـ C++ فيها صفة انها Case sensitivity يعني في فرق بين الأحرف الكبيرة والأحرف الصغيرة ، ف مسموولي اسمي متغير LINUX واسمي متغير ثاني linux

★ ممنوع يكون كلمة مدبوزة باللغة

- بينما ممكن يكون كلمة معرفة مسبقا مثل `cin / cout` , بس مش اشي كويش لأنه لو عملت برنامج فيه متغير اسمه `cout` , رح تذربط انت نفسك لما تكتب `cout` , هل قصدك عن أداة الطباعة ولا عن المتغير ! ، والكمبيوتر رح يدكي انه في `error` لما تستخدم المتغير بجملة الـ `cout` , لأنه هو بطل عارف إيه المتغير وأيابها الأداة

هون عرفت متغير من نوع integer (عدد صحيح) اسمه cout وذرت فيه قيمة 7 وما اعرض الكومبایلر ،

```
#include <iostream>
using namespace std;
int main()
{
    int cout;
    cout=7;
    cout<<cout;
}
```

بينها اعرض لما استخدمنه مع أداة ال cout

• Legal variables (identifiers) in C++:

- first
- conversion
- payRate

هاد السلايد في أمثلة على
أسماء مسموحة (legal) وأسماء
ممنوعة مع التعليل:

TABLE 2-1 Examples of Illegal Identifiers

Illegal Identifier	Description
employee Salary	There can be no space between employee and Salary.
Hello!	The exclamation mark cannot be used in an identifier.
one + two	The symbol + cannot be used in an identifier.
2nd	An identifier cannot begin with a digit.

- C++ variables have:
 - a name
 - a type
 - a value
- The variable exists as a result of a **declaration**
- A declaration is a syntactic structure that binds a name provided by the programmer with a specific type
- Type then **variable name** (or variable names separated by commas if there are more than one)
- Ends with a semicolon
 - `int Counter;`
 - `int variable1, account_balance, invoices;`

المتغير في لغة ال C++ إله 3 صفات، الاسم ودكتينا عنه والنوع والقيمة (بدددها النوع)

والمتغير ينوجد نتيجة لعملية اسمها **declaration** (إعلان)، وهاي العملية هي عبارة عن تركيب بربط الاسم الى بختاره المبرمج بنوع معين ،

بداية بنكتب النوع وبعددين بنكتب الاسم وبعدها الفاصلة المنقوطة ، واذا كنت بدبي اعمل اكثرب من متغير من نفس النوع بقدر اختصر حالياً واكتب اسم النوع مرة واحدة ، وفاصلة بين كل اسم (مثل المثال اللي بالسلايد)

-ملاحظة : لو عرفت متغير و م اعطيته قيمة رج تتخزن فيه قيمة عشوائية تلقائياً.

- The **type** is an **attribute** that uniquely defines which values can be stored inside the variable.
- To give a value to the newly declared variable, **assignment operator** is used.
- Examples:
 - `Counter = 1;` → assign 1 to Counter or Counter becomes 1.
 - `Result = 100+ 200;` → the new value of the variable *Result* will be the result of adding 100 to 200
 - `x= x + 1;` → Take the current value of the variable *x*, add 1 to it and store the result in the variable *x*. In effect, the value of variable *x* is **incremented** by one.

النوع هو الصفة يلي بتحدد القيمة يلي ممكن تخزنها جوا المتغير ، وعشان نعطي قيمة للمتغير بنستخدم إشارة المساواة = ، والتنفيذ يكون من اليمين لليسار (يتم اسناد قيمة الطرف اليمين للطرف الأيسر) وأي اخرى بلغة +++ يكون التنفيذ من اليسار لليمين

```
#include <iostream>
using namespace std;
int main()
{
int num ;
num=7;
7=num;
// غلط لأنه الطرف الأيسر لازم يكون متغير حتى اقدر احط فيه قيمة معينة
// تمام ما فيها مشكلة
int a,b,c,d; // 
int e=7,f=8; // نفس النوع
}
```

Data Types

- Data type: set of values together with a set of operations
- C++ data types fall into three categories:

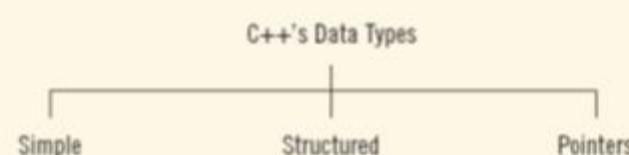


FIGURE 2-1 C++ data types

أنواع البيانات:-

التعريف يلي بالسلайд م الـ علقة بالحياة (أو أنا مش فاهمه)، تعريفها الصحيح أنه هي صفة للمتغير او للبيانات بتحدد للكومبيوتر كيف رح تستخدم هالمتغير (شو رح تخزن فيه) ويتقسم لثلاث أقسام زي مهو موضح بالصورة.

Simple Data Types

- Three categories of simple data
 - Integral: integers (numbers without a decimal)
 - Floating-point: decimal numbers
 - Enumeration type: user-defined data type

لو أخذنا أول نوع يلي هو **ال simple** رح برضو ينقسم لـ ٣ أقسام يلي هم **الاعداد الصديقة** , **integrals** والاعداد العشرية , **floating point** وآخر واحد نوع انت بتنيه وبتعدد صفاته اسمه **. Enumeration type**

Simple Data Types (cont'd.)

- Integral data types are further classified into nine categories:
 - char, short, int, long, bool
 - unsigned char, unsigned short, unsigned int, unsigned long

ال **integrals** **بنقسم ل ٩ انواع هيم**



FIGURE 2-3 Integral data types

int Data Type

- Examples:
 - 6728
 - 0
 - 78
 - +763
- Positive integers do not need a + sign
- No commas are used within an integer
 - Commas are used for separating items in a list

: Integer

نوع ال **integer** واختصاره **int** : هو عبارة عن أي عدد صحيح موجب أو سالب ، السالب لازم نحط قبله الاشارة السالبة ، بينما الموجب مش ضروري ، وم بصير يحتوي الرقم فواصل ، حكينا انه الفاصلة عشان تفصل العناصر بقائمة معينة ، مثل **int num,numb,number** ، ف هيكل declaration . declaration في جملة ال

bool Data Type

- bool type
 - Two values: true and false
 - Manipulate logical (Boolean) expressions
- true and false
 - Logical values
- bool, true, and false
 - Reserved words

: Boolean

نوع **Boolean** (منطقى) واختصاره **bool** بذن قيمتين فقط true ورمزها (1) , false ورمزها (0) ، وهاد النوع بتتحكم بالتعابير المنطقية (رح نتعرف عليها بدرس ال **(if clause)** الكلمات **bool,true and false** ، هي كلمات مدرجزة في لغة ال **++c** .

char Data Type

- Used for characters: letters, digits, and special symbols
- Each character is enclosed in single quotes
 - 'A', 'a', '0', '*', '+', '\$', '&'
- A blank space is a character
 - Written ' ', with a space left between the single quotes
- The smallest integral data type. **Computers store characters as numbers.**
- Every character used by computers corresponds to a unique number, and vice versa

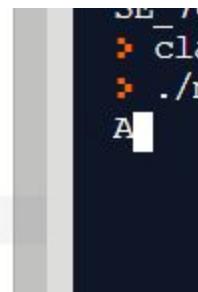
: Charchter

نوع character (رمز) و اختصاره **char**, ممكن يخزن خانة واحدة فقط, شو ما كانت, حرف أو رقم أو رمز اي زر بالKeyboard, وكل char يكون محاط بالsingle quotes (' ') مثل 'A'.

وبرضو ال space يلي بالكيبورد بنقدر ندكي انها char اذا دطينتها جوا ال single quotes , يعني بتعامل معها مثل اي رمز موجود .

- يجدر بالذكر إنه ال character هو أصغر نوع من حيث الحجم (رح نوضح الاحجام بالسلайд القادر)
- ومن المهم تعرف إنه الكمبيوتر يخزن أي شيء ع شكل أرقام , ف كل char في رقم مقابل إله مساوي إله , مثل لو بدي أخزن 'A' ف هاذ الحرف في رقم مقابل إله وهو 65 , وبالتالي رح تتخزن في الميموري على شكل 65.

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     char symbol=65;
5     cout<<symbol;
6 }
```



ف بنستنتج إنه **char 'A' = int 65** طبعا لو كنت معرف المتغير ك int رح يطبع 65 مش A .

TABLE 2-2 Values and Memory Allocation for Three Simple Data Types

Data Type	Values	Storage (in bytes)
int	-2147483648 to 2147483647	4
bool	true and false	1
char	-128 to 127	1

هاد الجدول بيبيلك كل قيمة بقدر كل نوع يخزنها , وحجمها بالذاكرة . شو يعني دجمها؟؟ , يعني أنا لما اعطي جملة declaration مثل

int n = 8;

الكومبييلر رح يدجز مكان بالرام بحجمه Byte-4 , ويخزن فيها قيمة ال 8 لكن بالنظام الثنائي (بتتعلم كيف يعني بالنظام الثنائي بمادة الديجيتال) , فعلينا ال 8 بت تحتاج 4byte , بس رح تدجزهم كلهم لأنه نوعها integer

بالنسبة للأقسام الثانية لل simple م اذكرت بالسليدات لكن اللي حاب يستزيد هاذ رابط ممتاز بشرحهم كلهم [الرابط](#) .

1.5 ASCII code

- A universal standard code implemented by (almost) all computers and operating systems all over the world
- ASCII (American Standard Code for Information Interchange) is the most widely used and nearly all modern devices (like computers, printers, mobile phones, tablets, etc.) use it. The code allows for 256 different characters.

م في اشي هاد السلايد بس دكي .. فقط مهم تعرف إنه ال ASCII هو إشي تم الاتفاق عليه بمثيل الأرقام يلي رح تمثل الرموز .
أمثلة :

ASCII Table

Character	Dec	Hex	Character	Dec	Hex
(space)	32	20	@	64	40
!	33	21	A	65	41
"	34	22	B	66	42
#	35	23	C	67	43
\$	36	24	D	68	44
%	37	25	E	69	45
&	38	26	F	70	46

Character	Dec	Hex
'	96	60
a	97	61
b	98	62
c	99	63
d	100	64
e	101	65
f	102	66
g	103	67

إلي بهمنا بس ال dec (العشري) ، طبعا هاد مش حفظ ، بس معلومات إنه كل حرف شو الترميز تاعه بالنظام العشري.

دكتينا إنه الرمز مساوي للرقم المقابل إله بال ASCII بالتالي لما ادكي لها ادكي `char x='A'+32` ، ف هو رح يبط .
يقيمة هالمتغير 97=32+65 وكونه نوعه char رح يخزن الرقم الخاص بيال 'a' .

Numbers and how computers see them

- The numbers handled by modern computers are of two types:
 - integers**, that is, whole numbers or those which are devoid of the fractional part,
 - floating-point numbers** (or simply **floats**) that contain (or are able to contain) the fractional part.
- Both of these kinds of numbers significantly differ in how they are stored in a computer memory and in the range of acceptable values.
- C++ language allows some **arithmetic operations** with numbers: add, subtract, multiply and divide.

الأرقام وكيف الكمبيوتر بتعامل معها!

الارقام بالكمبيوتر بتقسام لنوعين:-
(**integers**) الارقام الصريحة وهي يلي م فيها مكان لكسور العشرية.
النوع الثاني إلی قبل الفاصلة العشرية وإسمه **floating point** وهاد النوع بقبل كسور عشرية.

وهالنوعين بختلفوا بكيفية تخزينهم وبالد المسموح للقيم يلي ممكن تخزن فيهم .
وطبعا لغة الC++ بتسمح ببعض العمليات الرياضية ع هالنوعين مثل الجمع والطرح والضرب والقسمة.

Floating-Point Data Types

- C++ uses scientific notation to represent real numbers (floating-point notation)

TABLE 2-3 Examples of Real Numbers Printed in C++ Floating-Point Notation

Real Number	C++ Floating-Point Notation
75.924	7.592400E1
0.18	1.800000E-1
0.0000453	4.530000E-5
-1.482	-1.482000E0
7800.0	7.800000E3

The decimal point is essential to recognize floating-point numbers in C++.
4 is an **int** 4.0 is a **float**

لغة الC++ بتستخدم طريقة لتمثيل الأرقام العشرية (جوا الRAM) ، والتتمثيل يكون كالتالي

خانة وحدة قبل الفاصلة العشرية ، و 6 خانات بعد الفاصلة العشرية في الحرف E وبعد الحرف في الرقم يلي انت دركت الفاصلة العشرية بمقداره ، لو دركت لليمين بنقص الرقم واحد ولو دركت للشمال بزيادة الرقم واحد.

ويجدر بالذكر انه لازم يكون بالرقم فاصلة عشرية حتى اعتبره floating point number .

Floating-Point Data Types (cont'd.)

- **float**: represents any real number
 - Range: -3.4E+38 to 3.4E+38 (four bytes)
- **double**: represents any real number
 - Range: -1.7E+308 to 1.7E+308 (eight bytes)

On most newer compilers, data types **double** and **long double** are same

هاد النوع برضو يقسم لعدة أقسام
(**float** , **double** , **long double**)
وبالسلاليد موضع القيم يلي مسموح
تتخزن بكل قسم.

- وبأغلب الكومباینرات الـ **double** يعتبروا نوع واحد
والـ **long double** ياخذونه

فهمنا انه الارقام الها انواع (... ,int,float) ولما اعمل متغير انا بعده نوعه , طيب اذا دخلت رقم -مش متغير- ايش يكون نوعه ؟

- اذا كان الرقم من غير فاصلة فبكون نوعه **int**
- اذا كان مع فاصلة فبكون نوعه **double** - ادده انواع floating

اولوية الـ floating اعلى من الـ integral فلما اجمع او اضرب ... الخ بين integral مع floating فبكون الناتج floating

وإذا integral مع integral فالناتج دائمًا integral

```
int i, z;          float x, w;  
i = 10/4;           //i =2  
x = 10.0 / 4.0;    //x=2.5  
w = 10.0 / 4;      //w= 2.5  
z = 10.0 / 4;      //z=2
```

بالمثال يلي بالسلاليد في اربع متغيرات،
اثنين int (م فيه م مكان تخزين ارقام
عشريه) واثنين float

المتغير **i** (الارقام 10 و 4) int فتتخزن قيمة المتغير من غير فاصلة (**i**)
القيمة يلي رح تخزن فيه 2.5 لانه int مع float الناتج رح يكون بس كونه المتغير int رح يصير **lose data**
المتغير **x** بفاصله والارقام بفاصله فتتخزن بفاصلة عادي (**x**)
المتغير **w** قبل فاصلة والارقام - لانها اولى من int - فبخزنهم بفاصله (**w**)

مثال لما يكون المتغير من نوع بقبل الفاصلة وخرزنت فيه انجتر .
double c = 1; cout <<

رح يطبع 1 من غير صفر لانه (بالذاكرة بتتخزن بفاصله بس اثناء الطياعه ما بطبع 1.0 الا اذا عملت
setprecision الي رح تأخذوها باللب بعدين)

مثال قوي 3:

```
int i = (10/4)/2.5 ;
```

هون (10/4) مع int فجوابهم int يعني 2
بعدين 2/2.5 مع int double فجوابهم 0.8 - لانه الـ double اولويتها أعلى -
آخر اشي المتغير من نوع int فما باخذ الفاصله فالجواب 0 = i

- Maximum number of significant digits (decimal places) for `float` values is 6 or 7
- Maximum number of significant digits for `double` is 15
- Precision: maximum number of significant digits
 - Float values are called single precision
 - Double values are called double precision
- The **scientific notation** can be used to represent numbers that are very large or very small
 - Examples : 300000000 \square 3E8
 6.62607×10^{-34} \square 6.62607E-34
 - The exponent (the value after the "E") **must be an integer**.
 - The base (the value in front of the "E") **may or may not be an integer**.

C++ Programming: From Problem Analysis to Program Design, Fifth Edition

46

ال `float` يقدر بـ 6 أو 7 ارقام بعد الفاصلة ، بينما ال `double` يقدر بـ 15 .

precision في عندنا مفهوم جديد اسمه هو عبارة عن أكبر عدد من الأرقام يلي بعد الفاصلة العشرية

ممكن دد يسأل ، طيب ليش الكمبيوتر بدول الأوقام لهاي الصيغة (يللي بتخزن فيها ال floating بال RAM) ؟ ، لأنها ببساطة بتدول الرقم يلي بده مكان كبير لتمثيله ، لرقم بحاجة لمكان أقل . (بدل 3E4 بتخزن 30000

طبعا ال `exponent` لازم يكون `int` (لنه بمثل عدد خانات) بينما الرقم يلي قبله مش ضروري يكون `int`

- What happens when integer values converted into float values or vice versa?
- We can always transform from `int` into `float`.
 - For example:

```
int i ;
float f;
i=100;
f=i;
```

the value of the variable `i` will be 100.

the value of the variable `f` is 100.0

شو ممكن يصير لو حولت رقم من `int` ل `float` ؟
 ولا اشي © بس بتتضاف الفاصلة العشرية
 لقيمة الرقم بالذاكرة .

1.4 Floating-point numbers

- We can always transform from `float` into `int`, but it can lead to a **loss of accuracy**.

- For example:

```
int i ;
float f;
f=100.25 ;
i=f;
```

the value of the variable `i` will be 100

the value of the variable `f` will be 100.25

بينما لو حولت من `float` ل `int` راح أخسر القيمة يلي بعد الفاصلة العشرية (**lose data**)

1.4 Operators

- An operator is a symbol of the programming language, which is able to operate on the values
 - For example, an assignment operator is (=)
 - Used to assign values to variables $X = 4$
- Other operators available in C++ language associated with widely recognizable arithmetic operations.



العملية (**operator**) هي رمز في لغة البرمجة بقدر من خلاله اتحكم بالقيم ، على سبيل المثال اشارة المساواة = بمعنى من إني اسند قيمة للمتغيرات ، وبرضو في بلغة C++ العمليات الحسابية : جمع وضرب وقسمة وطرح إشارة الجمع + ، إشارة الطرح - ، إشارة الضرب * ، إشارة القسمة / ..

مثال :

```
1. #include <iostream>
2. using namespace std;
3. int main() {
4.     int a,b;
5.     float c,d;
6.     a=1.5; b=7.6;
7.     c=7.2 ; d=0.7;
8.     int e=c*b;
9.     float f = a-e;
10.    cout<<"E = "<<e<<endl;
11.    cout<<"f   ="<<f<<endl;
12. }
```

→ E = 50
f = -49

1.4 Division by zero

- This will result with a compilation error, runtime error or some message at runtime. As a general rule, you shouldn't divide by zero.
 - For example:

```
float x;
x = 1.0 / 0.0;
```
 - And the code:

```
float x,y;
x = 0.0;
y = 1.0 / x;
```
 - when you try to execute that code, the result of the operation is not a number. It's a special featured value named **inf** (as in **infinite**). Generally, this kind of illegal operation is a so-called **exception**.

القسمة على صفر

إذنًا تعلمنا نوع من الأخطاء يلي بكتشفها الكومبيوتر لما يكون عندي syntax error ، واسمه غلط ، وهالنوع الكومبيوتر بكتشفة وانت تكتب بالكود ، قبل عملية التنفيذ ، الآن راح نشوف نوع ثانى اسمه run time error ، وهالنوع بكتشفه الكومبيوتر أثناء تنفيذ البرنامج ، ومن الأمثلة عليه القسمة على صفر .

```
#include <iostream>
using namespace std;
int main() {
    int a=3,b=0;
    cout<<a/b;}
```



Runtime error

1.4 Unary minus (-)

- In "subtracting" applications, the minus operator expects two arguments: the left (a **minuend** in arithmetical terms) and right (a **subtrahend**).
 - For this reason, the subtraction operator is considered to be one of the binary operators, just like the addition, multiplication and division operators.
- The minus operator used as the unary operator, as it expects only one argument the right one.
 - For example: `int i,j;`

`i = -100;`
`j = -i;`
The variable `j` will be assigned the value of 100.

1.4 Unary plus (+)

- Its role is to preserve the sign. Take a look at the snippet
 - Although such a construction is syntactically correct, using it doesn't make much sense and it would be hard to find a good rationale for doing it.
 - For example:

`int i,j;`
`i = 100;`
`j = +i;`
The variable `j` will be assigned the value of 100.

عملية الطرح هي عملية ثنائية (**binary**) (بتوخذ طرفين المطروح والمطروح منه) في عندنا اشارة السالب **أحادية (Unary)** وهي يلي بتوخذ طرف واحد ع يمينها مثل المثال يلي بالسلайд ، ونفس الاشي الجمع عملية ثنائية ، بينما اشارة الموجب **أحادية (Unary)** (بتعطي اشارة للي بعدها) لكن م في داعي اكتبهما للرقم كونه عدم وجودها ووجودها برضو بشير لنفس الاشي (إنه الرقم موجب)

1.4 Remainder (%)

- Its representation in the C++ language is the % (percent) character.
- It's a binary operator (it performs **the modulo operation**) and both arguments cannot be floats.
- For example:

`int i,j,k;`

`i = 13;`
`j = 5;`
`k = i % j;`

- The `k` variable is 3 (because $2 * 5 + 3 = 13$).

باقي القسمة (modulo) واساراتها

بلغة الC++ هي هاذ الرمز **%** ، وهي عملية ثنائية (الها طرفين) ولازم الطرفين م يكونو من ال floating point numbers . يعني لازم من غير فاصله -

وبالطبع م بصير اعمل باقي قسمة لعدد ع صفر ، لانه لازم اعرف ناتج القسمة حتى اعرف الباقي ، يعني $13 / 5 = 2$ والباقي 3 ، ف انا احتجت اعرف الناتج حتى اجيب الباقي ، ف مثل 0.15% ، م يقدر اعرف اصلا 0.15 ، ف رح يعطيني **runtime error**

وبرضو لو عملت متغير من نوع float , double رح يعطيكي ايرور اذا استعملت معه % حتى لو مخزن فيه عدد صحيح

float x=1 ; cout <<x%2 ; <-- غلط

1.4 Priorities

- The hierarchy of priorities is the phenomenon that causes some operators to act before others is known as the **priority**.
- The C++ language precisely defines the priorities of all operators and assumes that operators of larger (higher) priority perform their operations before the operators with lower priority.
 - For example:

2 + 3 * 5

- The result will be 17.

الأولويات : كتعريف هي الخاصية
يللي بتسبي تنفيذ عملية قبل عملية
أخرى (ما علينا)

وأولويات لغة C++ نفسها أولويات
الرياضيات المعروفة .. فـ من الواضح
انه جواب المثال رح يكون 17

وفي حال كانت عندي الأولويات
متساوية ، بتتنفذ العمليات من اليسار
لليمين

مثال 6/2-5+4*7/6 الجواب رح يكون 7 (م تنسى 7/6 جوابها 1 لإنهم int)

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4
5     cout<<7/6*4+5-2;
6 }
```

وهاد الشكل فيه الأولويات للعمليات ..

+ -	unary
* / %	
+ -	binary

بينما لو دخلت عندي القوسـ ، رح تكون العملية يلي جوا القوس هي صاحبة الأولوية بغض النظر عنهاـ..

مثال جميـل للتدريب :

```
int i,j,k,l;
i = 100;
j = 25;
k = 13;
l = (5 * ((j % k) + i) / (2 * k)) / 2;
```

(20)

الجواب رح يكون 10

Expressions

- If all operands are integers
 - Expression is called an integral expression
 - Yields an integral result
 - Example: $2 + 3 * 5$
- If all operands are floating-point
 - Expression is called a floating-point expression
 - Yields a floating-point result
 - Example: $12.8 * 17.5 - 34.50$
- Mixed expression:
 - Has operands of different data types
 - Contains integers and floating-point
- Examples of mixed expressions:
 $2 + 3.5$
 $6 / 4 + 3.9$
 $5.4 * 2 - 13.6 + 18 / 2$
 - If operator has both types of operands
 - Integer is changed to floating-point

التعابير : Expressions

عملية أو أكثر على الأرقام (integral expression) إذا كانوا جميع الأطراف تأعات العمليات integers فبنسميه integral expression وأكيد الجواب int وإذا كانوا كلهم floating point numbers والجواب رح يكون فبنسميه floating point numbers وإذا كانوا خليط بين int و mixed floating point والجواب يكون expression (ذكرناها سابقا) floating point

Type Conversion

Cast operator: provides explicit type conversion
static_cast<dataTypeName>(expression)

EXAMPLE 2-9

Expression	Evaluates to
<code>static_cast<int>(7.9)</code>	7
<code>static_cast<int>(3.3)</code>	3
<code>static_cast<double>(25)</code>	25.0
<code>static_cast<double>(5+3)</code>	= <code>static_cast<double>(8) = 8.0</code>
<code>static_cast<double>(15) / 2</code>	= <code>15.0 / 2</code> (because <code>static_cast<double>(15) = 15.0</code>)
<code>static_cast<double>(15 / 2)</code>	= <code>15.0 / 2.0 = 7.5</code> = <code>static_cast<double>(7) (because 15 / 2 = 7)</code> = 7.0
<code>static_cast<int>(7.8 + static_cast<double>(15) / 2)</code>	= <code>static_cast<int>(7.8 + 7.5)</code> = <code>static_cast<int>(15.3)</code> = 15
<code>static_cast<int>(7.8 + static_cast<double>(15 / 2))</code>	= <code>static_cast<int>(7.8 + 7.0)</code> = <code>static_cast<int>(14.8)</code> = 14

C++ Programming: From Problem Analysis to Program Design, Fourth Edition

في operator يتسم بال تحول الداتا من نوع لنوع ثاني ..

وهيكل السنتاكس تأعها **static_cast<type>(value)** وممكن تكون ال value متغير او قيمة ثابتة .. الأمثلة يلي بالسلاليد كافية شافية شاملة كاملة D:

لا شيء ..

... ما لا تستطيع فعله ...

... يُؤثر على ...

.. ماتستطيع أن تفعله !

string Type

- Programmer-defined type supplied in ANSI/ISO Standard C++ library
- Sequence of zero or more characters
- Enclosed in double quotation marks
- Null: a string with no characters
- Each character has relative position in string
 - Position of first character is 0
- Length of a string is number of characters in it
 - Example: length of "William Jacob" is 13

```
string x;  
x= "Hello World20 ?"; //15 characters
```

في نوع Data ثانوي واسمه **string**

. " " double quotation عن (صفر كراكتر أو أكثر) ولازم يبدأ وينتهي بال (شفناه قبل هيئ) وهو عبارة

في عندنا إشي اسمه NULL (كله كابتل) وهو string يلي فيه **zero char** يعني ولا اشي .

وكل كراكتر داخل ال string إله مكان خاص فيه , وترقيم هالاماكن ببلش من عند الصفر , يعني أول char داخل ال string تكون الموضع تاعه 0.

وطول السترينج تكون عدد ال char يلي فيه (بتبلش تعد من واحد مش من صفر) .

مثلا عندي " string s="JUST32 , إذا طلب من ال position 3 تكون الجواب T ، وطول ال string كامل 6 .

1.4 Increment Operator (++)

- Used to increment a variable by one.
 - This is often done when we're counting something .
 - For example:

```
int Counter;  
Counter = 0;  
Counter = Counter + 1;  
  
• We achieve the same effect in a shorter way:  
Counter++;
```

(Increment Operator) وهي بإختصار
بتعمل زيادة على المتغير بمقدار 1 .

مثلا أنا بعد أشياء , بدل كل م بدبي ازيد أكتب
counter ++ counter=counter+1
والجملتين هذول متكافآت تماماً .

طبعاً بصير استخدمها مع ثوابت مش
متغيرات , لانه لو بدبي ادكي 7++ كانني
دكينت 1=7+1=8 وطبعاً قيمة ال 7 لتساوي
لذلك هاي العملية مش مسموحة غير
للمتغيرات (هي وأي عملية بتعمل على
تغيير القيم)

1.4 Decrement Operator (--)

- Used **to** decrease the value of a chosen variable by one.
 - This is often done when we're counting something .
 - For example:

```
int Counter;  
Counter = 10;  
Counter = Counter - 1;
```

- We achieve the same effect in a shorter way:

Counter--;

های ال operator نفس ال increment بس انها بتتفص 1. ☺

ولازم نعرف إنه هاي ال increment أو ال decrement ممكن يكونوا بشكليين ، الأول اسمه **prefix (سابقة)** والثاني اسمه **postfix (لاحقة)** ورح نوضح الاختلاف بينهم من خلال مثال **حارق خارق متغير**:

```
#include <iostream>
using namespace std;
int main() {
    int m=5 , o , h;
    o=m++; // postfix increment
    cout<<"o = "<<o<<endl;
    h=++m; // prefix increme
    cout<<"h = "<<h<<endl;
    cout<<"m after postfix decrement ="<<m--<<endl;
    cout<<"m after prefix decrement ="<<--m<<endl;
}
```

```
final)
> clang++-7 -pthread -o main ma
> ./main
o = 5
h = 7
m after postfix decrement =7
m after prefix decrement =5

```

عندنا متغيرات m , o , h وقيمة $m=5$, دكينا انه ال $++o$ ف المفروض انه $o=6$, لكن o ضلت 5 ليش !!
لعله أولوية ال **post ضعيفة جدا**, أقل من أولوية المساواة ومن أي أولوية أخرى, ف تم اسناد قيمة ال 5 لل o وبالتالي $o=5$ وال $m=6$ وال $o=5$ لا يزداد .

وحكينا الـ **m** و **h=++m** ف هل رح يعمل إسناد قبل مزيد الـ 1 ولا بعد ؟
العلويه في الـ **pre عاليه جدا** , ف بنفذها قبل ما ينفذ أي عملية , ف زاد على قيمة **m** وصارت 7 وبعدين
أي طلاقا **b**

پهای تیک الارکیات بعد م دخانا علیهم الی pre:inc/dec

++ -- + -	unary
* / %	
+ -	binary
=	

1.4 Shortcut operators

- In the C++ language there is a short way to write the operators.
- If op is a two-argument operator and the operator is used in the following context:

variable = variable **op** expression;

It can be simplified and shown as follows:

variable **op** = expression;

- For example:

`i=i*2; → i *= 2 ;`

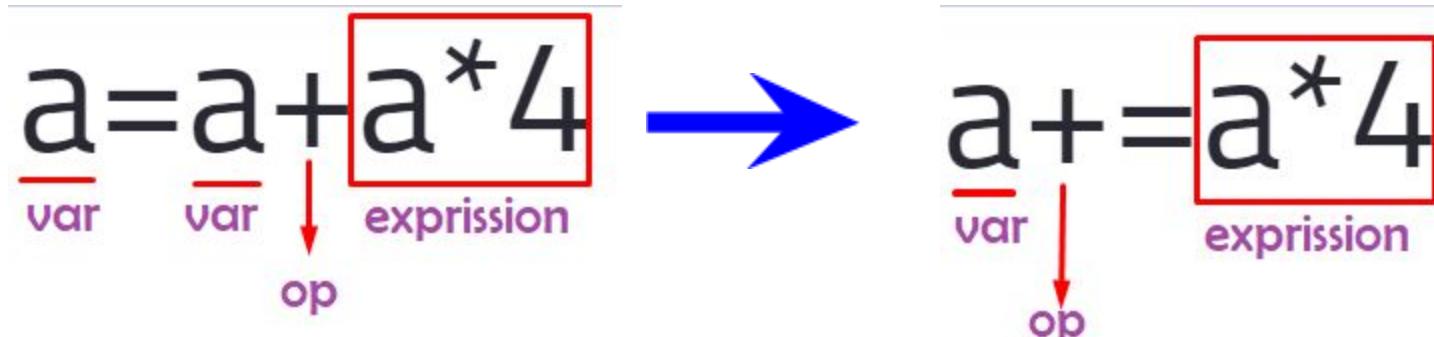
`counter = counter + 10; → counter +=10;`

العمليات المختصرة (**shortcut operators**) : هي عبارة عن طريقة لكتابة العمليات بشكل مختصر.

لكن لازم العملية يتطبق عليها شرطين : الاول انها تكون binary (الها طرفين مثل الجمع او الطرح)

الثاني إنها تكون مع هاي الصيغة :

variable **operator** = expression



i = i + 2 * j;	i += 2 * j;
Var = Var / 2;	Var /= 2;
Rem = Rem % 10;	Rem %= 10;
j = j - (i + Var + Rem);	j -= (i + Var + Rem);

1.5 Character and Literals

- **char is a type**
- **literal is a symbol that uniquely identifies its value**
 - Examples:
 - char Character = 'A'
 - char A = 100
 - Character: this is not a literal; it's a variable name
 - 'A' : this is a literal; when you look at it you immediately know its value
 - 100 : it's a literal, too (of type *int*)
 - 100.0 : it's another literal, this time of the *float* type
 - j + 100 : this is a combination of a variable and a literal joined together with the + operator; such a structure is called an expression.

رح نعرف بهاد السلايد مصطلح جديد اسمه **literal** (**القيمة الثابتة**) : وهو عبارة عن الرمز يلي بدد القيمة .

يعني مثلاً بدي أعتبر عن قيمة مقدارها سبعة ف الرمز يلي بمكنني من التعبير عن هاي القيمة هو الرمز 7 ف الـ 7 تكون literal .

: أمثلة

```
int num=7 ;  
char sym='h';  
string team ="Jsne"  
string another_team ="softwarian"  
char m ='a'+i;
```

بهاي الأمثلة عندي name of variable وعندى ال data type . بينما آخر مثال كانت القيمة literal مش تكون كلها ثوابت حتى اعتبرها expression

- The C++ language uses a special character backslash(\)
- it named as **escape character** because by using the \ we escape from the normal meaning of the character that follows the slash
- You can also use the escape character to escape from the escape character.
 - For example:

Character = '\\';

لغة ال c++ بيستخدم رمز خاص ال \ backslash ويسمى **escape character** الهرب : لانه بغير (يهرّب من) الدلالة الطبيعية للـ رح ينكتب بعده)

واحنا برضو بنقدر نرجعه لك رمز عادي بس نكتبه مرتبين ورا بعض .

	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

هذا السلايد فيه بعض الرموز ودلائلهن
 الأول بعمل سطر جديد ، الثاني بيمني مقدار معين من الفراغات ، الثالث بمسح الحرف الي قبله ،
 الرابع يرجع المؤشر لأول السطر ، الخامس حكينا عنه ، السادس بتقدر تطبع من خالله ال (')
 والسابع بتقدر تطبع من خالله ال (") **double quotation**

رح نوضح بمثال تفصيلي

```

1 #include <iostream>
2 using namespace std;
3 int main() {
4     cout<<"Jsne"<<"\n"<<"team\n"; //نزل سطر
5
6
7     cout<<"softwarian"<<"\t"<<"team\n"; // وبعد مقدار معين من الفراغات
8
9     cout<<"osamaa\b momani\n"; // بنوصحها تحت
10
11
12    cout<<"mhmd \rosama\n"; // رح يكتب اول اربع احرف ووراهم فراغ بعدين رح يرجع ع اول السطر
13    /*ويكتب فوق الاحرف الي كانت موجودة اسم اسامه
14    cout<<'''<<endl; // أول طريقة مثان نطبع
15    cout<<'\''<<endl; // ثان طريقة
16    cout<<"\\"<<endl; // طريقة اطبع
17
18 }
```

بالنسبة للسطر التاسع ، بس كتبنا \ الممؤشر رجع خطوة لورا وكتب الحرف الي بعد ال \b (كان فراغ بالمثال) على المكان الي موجود عليه حاليا (كان حرف a) بالمثال .

```

Jsne
team
softwarian team
osama momani
osama
.
.
"

```

وهاي شاشة الاوتبوت:

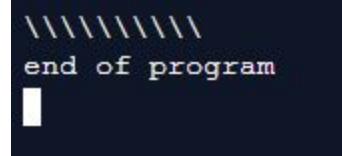
مثال :-

cout << "JavaScript\rC++" ; --> output : C++aScript

- `\a` : as in alarm if you send this character to the screen, you'll hear a short beep.
 - `\0` : called **null** (none) is a character that does not represent any character.

في \a وهو رمز بصدر صوت
(بيب) لما تطبعه ع الشاشة
ودىينا عن الـNUL ومطلوب
تعرف إنه هاذ رمزه: \0

م رح يطبع أي اشي مكان الـ h
وهاي الاوتبوت



رجح ننتقل لموضوع مهم جداً وهو كيف تتخذ قرار في لغة الـ C++ بناءً على شرط معين ..

1.6 Relational operators

- To ask questions, the C++ language uses a set of very special operators called **relational operators**.
 - The result from expressions that use relational operators will be true or false.
 - A condition is represented by a logical (Boolean) expression that can be true or false
 - Relational operators:
 - Allow comparisons
 - Require two operands (binary)
 - Evaluate to true or false

بالبداية رح نتعلم كيف نسأل اسئلة بلغة ال C++ :

رَحْ نَسْتَخْدِمُ مَجْمُوعَةً مِنْ
الْمُهَاجِراتِ وَالْمُسَافِرَاتِ

relational operators

ونتيجة هذه العمليات يمكن أن تكون `true` أو `false` وتحتاج إلى طرفين (binary operator) يسمح بالمقارنة بين أطرافها.

والشرط يلي بحدد النتيجة
بنعبر عنه ب تعبير منطقى
boolean expression
(بنفسه شىء هو بعثا)

وهي كل عملية والوصف تابعاً.

Operator	Description
<code>==</code>	equal to
<code>!=</code>	not equal to
<code><</code>	less than
<code><=</code>	less than or equal to
<code>></code>	greater than
<code>>=</code>	greater than or equal to

بعض الأمثلة على ارقام :-

- You can use the relational operators with all three simple data types:

- `8 < 15 evaluates to true`
- `6 != 6 evaluates to false`
- `2.5 > 5.8 evaluates to false`
- `5.9 <= 7.5 evaluates to true`

TABLE 4-2 Evaluating Expressions Using Relational Operators and the ASCII Collating Sequence

Expression	Value of Expression	Explanation
<code>' ' < 'a'</code>	<code>true</code>	The ASCII value of ' ' is 32, and the ASCII value of 'a' is 97. Because 32 < 97 is <code>true</code> , it follows that ' ' < 'a' is <code>true</code> .
<code>'R' > 'T'</code>	<code>false</code>	The ASCII value of 'R' is 82, and the ASCII value of 'T' is 84. Because 82 > 84 is <code>false</code> , it follows that 'R' > 'T' is <code>false</code> .
<code>'+' < '*'</code>	<code>false</code>	The ASCII value of '+' is 43, and the ASCII value of '*' is 42. Because 43 < 42 is <code>false</code> , it follows that '+' < '*' is <code>false</code> .
<code>'6' <= '>'</code>	<code>true</code>	The ASCII value of '6' is 54, and the ASCII value of '>' is 62. Because 54 <= 62 is <code>true</code> , it follows that '6' <= '>' is <code>true</code> .

الآن رح نشوف امثلة على الرموز

عملية المقارنة بتتم عن انه بدول كل رمز لل ascii تاعه وبقارنهم ل ارقام .

يقدر بالذكر عند اخر مثال إنه **6 ك int مختلف** **اختلف كامل عن 6** لنه الأخيرة هي رمز ascii للرقم مش الرقم نفسه وهذا الرمز الـ 54 خاص فيه وهو .

Relational Operators and the string Type

- Relational operators can be applied to strings
- Strings are compared character by character, starting with the first character
- Comparison continues until either a mismatch is found or all characters are found equal
- If two strings of different lengths are compared and the comparison is equal to the last character of the shorter string
 - The shorter string is less than the larger string

الآن رح نشوف ال relational operators على ال **string**

عملية المقارنة بتتم كال التالي : بقارن ال strings يلي موجودة جوا ال characters واحد واحد , عند اول اختلاف بين ال characters بيتبذل قراره (اكبر او اصغر حسب الحرف) , وادا م لقا اي اختلاف تكون ال string اللطول هو الأكبر , وادا م لقا اي اختلاف وكانو نفس الطول ف تكونو متساوين .

```
#include <iostream>
using namespace std;
int main() {
    string s="mhmd";
    cout<<boolalpha<<(s=="mhmd1")<<endl;
    cout<<(s=="mhmd1")<<endl;
}
```

ال **boolalpha** حتى يطبع `0` او `1` ل `false` او `true` بدونها رح يطبع `0` او `1` ، ال boolean expression هاذ يلي بين قوسين ، ولازم دائمما يكون بين أقواس .

```

string str1 = "Hello";
string str2 = "Hi";
string str3 = "Air";
string str4 = "Bill";

```

TABLE 4-3 Evaluating Logical Expressions with string Variables

Expression	Value	Explanation
str1 < str2	true	str1 = "Hello" and str2 = "Hi". The first characters of str1 and str2 are the same, but the second character 'e' of str1 is less than the second character 'i' of str2. Therefore, str1 < str2 is true.
str1 > "Hen"	false	str1 = "Hello". The first two characters of str1 and "Hen" are the same, but the third character 'l' of str1 is less than the third character 'n' of "Hen". Therefore, str1 > "Hen" is false.
str3 < "An"	true	str3 = "Air". The first characters of str3 and "An" are the same, but the second character 'i' of "Air" is less than the second character 'n' of "An". Therefore, str3 < "An" is true.
str1 == "hello"	false	str1 = "Hello". The first character 'H' of str1 is less than the first character 'h' of "hello" because the ASCII value of 'H' is 72, and the ASCII value of 'h' is 104. Therefore, str1 == "hello" is false.

في اختلاف و الـ **a** أكبر من **e**

في اختلاف و الـ **a** أكبر من **n**

في اختلاف و الـ **a** أكبر من **i**

في اختلاف لأنه **h** لتساوي **H**

Logical (Boolean) Operators and Logical Expressions

- Logical (Boolean) operators enable you to combine logical expressions

TABLE 4-5 Logical (Boolean) Operators in C++

Operator	Description
! ← unary	not
&& ← binary	and
← binary	or

الآن رح نشوف نوع ثان من الـ **operators** وهو الـ **logical أو boolean (المنطقي)** وهما العمليات بتخليني اعمل تركيب بين الـ **logical expression** (يلي بحتووا **relational operator**)

في 3 عمليات وموضدين بالصورة , كل عملية شو رمزها وشو اسمها

الـ **not** بتعكس الـ **true** ل **false** (والعكس) والـ **operator** واحد (unary operator)

الـ **and** إلـ **ها طرفين** (binary operator) والنتاج يكون **true** إذا كانواا الطرفين **true** , غير هيك يكون **false**

افراح آلـ**المنطق** XD

EXAMPLE 4-2

Expression	Value	Explanation
<code>! ('A' > 'B')</code>	true	Because <code>'A' > 'B'</code> is false, <code>! ('A' > 'B')</code> is true.
<code>! (6 <= 7)</code>	false	Because <code>6 <= 7</code> is true, <code>! (6 <= 7)</code> is false.

EXAMPLE 4-3

Expression	Value	Explanation
<code>(14 >= 5) && ('A' < 'B')</code>	true	Because <code>(14 >= 5)</code> is true, <code>('A' < 'B')</code> is true, and true && true is true, the expression evaluates to true.
<code>(24 >= 35) && ('A' < 'B')</code>	false	Because <code>(24 >= 35)</code> is false, <code>('A' < 'B')</code> is true, and false && true is false, the expression evaluates to false.

EXAMPLE 4-4

Expression	Value	Explanation
<code>(14 >= 5) ('A' > 'B')</code>	true	Because <code>(14 >= 5)</code> is true, <code>('A' > 'B')</code> is false, and true false is true, the expression evaluates to true.
<code>(24 >= 35) ('A' > 'B')</code>	false	Because <code>(24 >= 35)</code> is false, <code>('A' > 'B')</code> is false, and false false is false, the expression evaluates to false.
<code>('A' <= 'a') (7 != 7)</code>	true	Because <code>('A' <= 'a')</code> is true, <code>(7 != 7)</code> is false, and true false is true, the expression evaluates to true.

في إشي بدننا ننوه عليه : ال and operator إذا كان واحد من اطرافها يساوي 0 , ف أكيد الجواب صفر , ف م بتشوف الطرف الثاني أبدا ولا بتنفذه وهماي إسمها **Short-circuit evaluation** . وال OR نفس اللي إذا كان واحد منهم يساوي 1 ف أكيد الجواب 1 .

مثال :

```
#include <iostream>
using namespace std;
int main() {
int m=1;
int o=0;
cout<<boolalpha<<(o&&-m)<<endl;// عمل الديكرمنت
cout<<m<<endl;
cout<<boolalpha<<(m&&o--)<<endl;/
/* كمل وحسب الطرف اليسرى
cout<<o<<endl;
}
```

output:

```
false
1
false
-1
[]
```

بس شاف إنه أول طرف بساوي صفر دغري طلع و م عمل الديكرمنت هون كان أول طرف 1 ف لسا م قدر يحدد الناتج */

ملاحظة : الاصل بال boolean operator استعمل معهم متغيرات من نوع bool بس بصير استعمل معهم اي نوع وبنكون كل القيم بتمثل 1 , true , ما عدا ال 0 بتمثل false , يعني

6.2 && -5 هي نفسها ('A' && 'B') = 1 ونفسها (1&&1) الخ ..

صار في عدنا عدة أنواع من الـ **operators**

1-arithmetic

2-relational

3-boolean

ف لازم نشوف الأولية بينهم كلهم ، وهي على النحو التالي :

TABLE 4-9 Precedence of Operators

Operators	Precedence
!, +, - (unary operators)	first
*, /, %	second
+, -	third
<, <=, >=, >	fourth
==, !=	fifth
&&	sixth
	seventh
= (assignment operator)	last

والآن رح نتعرف كيف نبني شرط في لغة الـ C++.

- There are several variants of **conditional instruction**
 - One-Way Selection
 - Compound (Block of) Statements
 - Two-Way Selection
 - Multiple Selections: Nested **if**
 - Comparing **if...else** Statements with a Series of **if** Statements
- In this chapter we will focus on the first and the second forms

في أربع طرق إحنا بهاد التشاير رح نوخد أول ثنتين

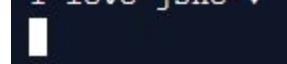
نبلاش بال **one way selection**

بكون عندي شرط ، إذا تحقق (كان ناتج التعبير المنطقي true) رح ينفذ جواب الشرط (أي جملة برمجية بدئياً يها)

```
#include <iostream>
using namespace std;
int main() {
    if(4>3)
        cout<<"I love jsne ♥"<<endl;
}
```



I love jsne ♥

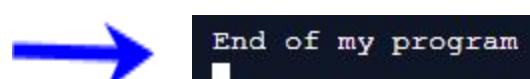


يعني عندى شرط وعندى أدلة شرط وعندى جواب شرط وهذا توضيح

وهماي الأوببوت (لأنه الشرط تتحقق نفذ الجملة البرمجية)

إذا م تحقق م رح ينفذها.

```
#include <iostream>
using namespace std;
int main() {
    if(4==3)
        cout<<"I hate jsne"=<<endl;
        cout<<"End of my program "<<endl;
}
```



هون مانفذ جواب الشرط بس نفذ

العبارة الي تحت لأنها ما

الها دخل بال **if**

في إشي لازم تعرفه وعادي بجيروا عليه اسئلة في الامتحانات والي هو اذا بعد جملة ال if كتبت فاصلة **منقطة** ، ف هاي الفاصلة بتنتهي عمل ال if يعني زي كإنك دكيت اذا الشرط تتحقق م بدي تعمل اشي مثل :

EXAMPLE 4-12

Consider the following C++ statements:

```
if (score >= 60);           //Line 1
    grade = 'P';            //Line 2
```

Because there is a semicolon at the end of the expression (see Line 1), the **if** statement in Line 1 terminates. The action of this **if** statement is null, and the statement in Line 2 is not part of the **if** statement in Line 1. Hence, the statement in Line 2 executes regardless of how the **if** statement evaluates.

هون 2 line رح يتنفذ بغض النظر عن جملة ال if ، لأنها الفاصلة المنقطة ووقفت شغل ال if.

ولازم ننوه إنه لازم يكون في جواب للشرط ، يعني لو بدق اذا تتحقق الشرط م تعمل اشي ف لازم تحط فاصلة منقطة (هي بتكون جواب الشرط) و م بصير ترکه فاضي.



ولازم يكون في شرط (م بصير تخلی الاقواس فاضية).
ممكن تكتب بالشرط أي رقم وبعتبره true إذا ما كان صفر.

```
if (000) cout << "Zero --> False";
if ('Z') cout<<"Otherwise it is True "
;
```

Otherwise it is True

مثال :

Confusion Between == and =

- C++ allows you to use any expression that can be evaluated to either **true** or **false** as an expression in the **if** statement:

```
if (x = 5)
    cout << "The value is five." << endl;
```

- The appearance of = in place of == resembles a *silent killer*
 - It is not a syntax error
 - It is a logical error

لازم تنتبه وتفرق بين ال == (هل يساوي) وال = (المساواة) ف لو كتبت = بدل == رح توقع ب والي رح بصير إنه رح يسند القيمة للمتغير والشرط رح يكون **true**

مثال :

```
#include <iostream>
using namespace std;
int main()
{
int n;
if(n=7)
cout<<"hello its me"<<endl;
cout<<n<<endl;
}
```

hello its me
7

والآن رح نشوف النوع الثاني Compound (Block of) Statements.

```
{  
    statement1  
    statement2  
    :  
    :  
    statementn  
}
```

When we have to execute conditionally more than one instruction, we need to use the braces { and } which create a structure known as a **compound statement** or (much simpler) a **block**.

The compiler treats the block as a single instruction.

- A compound statement is a single statement

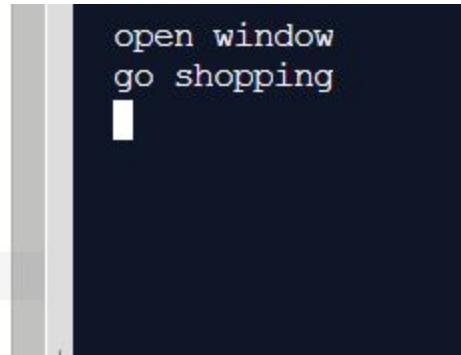
إحنا تعلمـنا إنـه إذا تنفذ شـرط أـنـفذ جـملـة ، طـيـب لو بـدـي أـنـفذ أـكـثـر مـن جـملـة إـذـا تـحـقـق الشـرـط شـو أـعـمـل !

يعـني مـثـلاً إـذـا كانـ الجـوـ بـكـرة جـيد رـجـع {أـروـح لـلـسـوق وـافـتـح الشـبـابـيك}

فـأـنـا رـيـطـتـ جـمـلـتـيـن بـشـرـطـ وـاحـدـ ، فـالـحلـ بـكـونـ إـنـي اـحـطـهـمـ بـإـشـيـ اسمـهـ block

وبـيـدـأـ بـهـادـ الرـمـزـ {ـ وـ بـنـتهـيـ بـهـادـ }ـ وـ الـكـوـمـبـاـيـلـرـ بـتـعـامـلـ مـعـ الـبـلـوـكـ كـأـنـهـ جـمـلـةـ وـبـدـءـ .

```
bool good_weather = true ;  
if (good_weather)  
{  
cout<<"open window" << endl;  
cout<<"go shopping " << endl;  
good_weather=false ;  
}
```



مـثـالـ :

هـوـنـ نـفـذـ كـلـ إـلـيـ جـواـ الـبـلـوـكـ مـرـهـ وـدـدـهـ عـالـتـرـتـيـبـ

خـلـيـنـاـ نـشـوـفـ تـطـبـيقـ عـمـلـيـ عـلـىـ الـ if~statement~ بـدـلـ الـمـثـلـةـ يـاـ كـاـهـاـ اـسـمـةـ وـمـحـمـدـ وـجـسـنـيـ D:

```
#include <iostream>  
using namespace std;  
int main() {  
int mark;  
cout<<"Enter your mark: "<< endl;  
cin>>mark;  
if(mark>100 || mark<0)  
cout<<"invalid input" << endl;  
if(100>=mark&&mark>=50)  
cout<<"pass" << endl;  
if(mark<50 and mark>=0 )  
cout<<"failed" << endl;  
}
```

هـادـ بـرـنـامـجـ بـتـعـطـيـهـ عـلـمـتـكـ وـبـدـكـيـلـكـ إـذـاـ الـعـلـمـةـ صـحـ أـوـ غـلـاطـ وـبـدـكـيـلـكـ نـاجـحـ وـلـاـ رـاسـبـ .

بـالـنـسـبةـ لـلـجـمـلـةـ يـاـ بـسـطـرـ 6ـ ،ـ رـحـ نـشـرـهـاـ بـالـتـفـصـيلـ كـمـانـ شـوـيـ ،ـ بـسـ بـدـيـ تـفـهـمـ مـنـهـاـ إـنـهـاـ بـتـطـلـبـ مـنـ الـمـسـتـخـدـمـ مـعـلـومـاتـ ،ـ فـ مـثـلاـ هـوـنـ طـلـبـتـ مـنـهـ قـيـمـةـ الـعـلـمـةـ

وـسـطـرـ 9ـ مـمـكـنـ وـاـدـ بـدـكـيـ خـلـيـ الشـرـطـ if(50>= mark>=100)ـ ،ـ وـهـادـ غـلـطـ كـثـيرـ نـاسـ بـتـوـقـعـ فـيـهـ ،ـ لـازـمـ تـتـذـكـرـ إـنـهـ جـوابـ الـr~elational operatorـ هـوـ عـبـارـةـ عـنـ 1ـ أـوـ 0ـ ،ـ فـ مـثـلاـ لـوـ كـانـتـ الـعـلـمـةـ 70ـ ،ـ رـحـ يـشـوـفـ إـذـاـ الـ 50ـ أـكـبـرـ مـنـ أـوـ يـسـاـوـيـ الـ 70ـ وـالـجـوابـ f~alseـ فـ رـحـ يـكـونـ 0ـ ،ـ بـعـدـيـنـ رـحـ يـشـوـفـ إـذـاـ الـ 0ـ أـكـبـرـ مـنـ أـوـ يـسـاـوـيـ الـ 100ـ وـرـحـ يـكـونـ الـj~o~a~b~ فـ مـ رـحـ يـطـبـعـ p~ass~ ،ـ لـهـيـكـ إـذـاـ كـانـ الـشـرـطـ فـتـرـةـ لـازـمـ تـكـتـبـ بـهـاـ الصـيـغـةـ وـبـاـسـتـخـدـامـ orـ andـ

وـبـالـنـسـبةـ لـسـطـرـ 11ـ ،ـ أـنـاـ كـتـبـتـ a~nd~ بـهـالـطـرـيـقـةـ مـشـانـ اـحـكـيـلـكـ إـنـهـ فـيـ بـعـضـ الـكـوـمـبـاـيـلـرـاتـ (ـخـصـوصـاـ الـوـنـلـلـيـنـ)ـ بـقـبـلـوـ هـالـكـتـابـةـ ،ـ وـبـقـبـلـوـ تـكـتـبـ الـ ||~not~ هـيـكـ or~

وـهـادـ رـابـطـ الـk~o~d~ .

الآن رح نشوف موضوع جدید المفروض اشرح قبل (بس هيك ترتيب السليفات)

1.7 Input and output

- When data moves in the direction of the human (user) to the computer program, it's called the **input**
 - `cin` stream, used with the `>>` operator to insert data.
 - The `>>` operator itself is sometimes referred to as an **insertion operator**.
- The data transferred from the computer to the human, is called the **output**.
 - `cout` stream, used with the `<<` operator to output data.
 - The `<<` operator itself is sometimes referred to as an **insertion operator**.

يبركيك المعلمات (DATA) لما تنتقل من المستخدم للبرنامج بنسمها مدخلات (inputs) وبنستخدم أداة ال `cin` وال extraction operator (إسماها `>`) : اداة الادخال * السلايد مخبرط لعمام ها العملية.

وال `cin` بتحول الداتا يلي يتدخلها الى لغة الآلة.

وال Data يلي بتنتقل من الحاسوب (output) للمسخدم تسمى مخرجات (output) وبنستخدم أداة ال `cout` وال insertion operator `<<` (إسماها `<<`) : اداة الإدراج لعمام ها العملية.

To print the value of an integer variable we need to send it to `cout` stream through `<<` operator

You can connect more than one `<<` operator in one `cout` statement and each of the printed elements may be of a different type and a different nature

```
int herd_size = 123;
cout << "Sheep counted so far: " << herd_size;
```

```
int square_side = 12;
cout << "The square perimeter is: " << 4 * square_side;
```

```
char Char = 'X', Minus = '-';
float Float = 2.5;
cout << Char << Minus << Float;
```

رج نشرح ال output

مشان تطبع أي قيمة لازم تبعثها لل `cout` من خلال ال extraction operator `<<` إنك تستخدم أكثر من extraction operator في نفس جملة ال `cout` كل وحدة منهم بتطبع يلي ببعثه بعدها ، وممكن الأشياء يلي ببعثها لجملة ال `cout` تكون مختلفة عن بعض يعني var literal أو `var` ، وال يمكن يكون أي نوع.

مثال :

```
#include <iostream>
using namespace std;
int main()
{
    int a=0;
    int b=4;
    cout<<"a ="<<a<<'t'<<"b ="<<b;
```

a =0 b =4

وهاي الاوتبورت

بالمثال هاذ ارسلت للطباعة من نوع string literal وبرضو متغير float char literal g int

- You can use conversion type with `cout` stream.
- A conversion can be done using the syntax: `(newtype)expr`
 - Changes the type of the `expr` expression into the `newtype` type.
- Using this conversion we can see the ASCII code of any character stored within a `char` variable and vice versa, or see a character whose ASCII code is placed inside an `int` variable.
 - For example:

```
char Char = 'X';
int Int = Char;
cout<<Char<< " <<(int)Char<< " <<Int<< " <<(char)Int;
```

 - This snippet outputs the following text to the screen: X 88 88 X

بنقدر نستخدم تدويل النوع للمتغير مع جملة الـ `cout`.
 في طريقتين لتدويل المتغير الأولى أخذناها `static_cast<type>(value)` ورح نشوف طريقة اسهل واسرع ورح نشوف السنتاكس الها بمثال.

```
#include <iostream>
using namespace std;
int main() {
    char n='n';
    cout<<(int)'n'<<endl;
    cout<<static_cast<int>(n);
```



```
110
110
```

مشان اخلي المؤشر ينزل سطر عندي طريقتين الأولى يلي تعلمناها في ال escape characters . var gl literal يلي بده تدولها وهيك بتقدر تشو夫 أي ascii لأي رمز موجود عندك ع الكيبورد .

`cout << "1\n2" << endl << "3\n";` مثال :

رح تطبع :

1

2

3

بقدِّرِ الْكَدِّ تَلَسُّبُ الْعَالَى ..

.. وَمَنْ طَلَبَ الْعَالَى سَهَّرَ الْلَّيْلَى وَمَنْ رَامَ الْعَالَى مِنْ غَيْرِ كَدِّ ..

.. أَضَاعَ الْعُمُرَ فِي طَلَبِ الْمَحَالِ تَرَوْمُ الْعَزَّ ثُمَّ تَامُ لَيَلَ ..

.. يَغْوِصُ الْبَحْرُ مِنْ طَلَبِ الْأَرَابِيِّ ..

1.7 Input

- Using **cin** stream with the operator **>>**; C++ language program get data from a human and store it in variables.
- C++ program can ask the user to enter a value from the keyboard and the program stores it in a specified variable
 - For example: `cin>>MaxPrice;`
- A difference between cout and cin streams is that the argument for *cout* may not be a variable. It can also be an expression or constants.
 - For example: `cout << 2 * i;`
 - But using an input stream, we need to **explicitly specify the variable** that can store the data entered by the user.

شرح الـ **input**

بنستخدم الـ **cin** لنأخذ معلومات من المستخدم ونخزنها في فاربيل ، والفرق بين الـ **cin** والـ **cout** : إنه زي م شفنا الـ **cout** ممكن تأخذ ع طرفها **literal** او **var** بينما الـ **cin** **ممنوع** **var** **طرفها غير**

```
#include <iostream>

using namespace std;

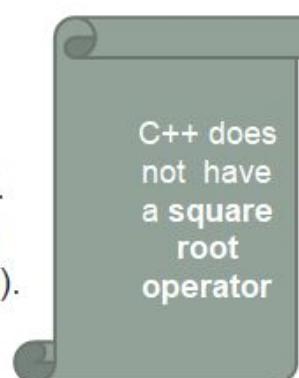
int main(void)
{
    int value,square;

    cout << "Give me a number and I will square it!\n";
    cin >> value;
    square = value * value;
    cout << "You've given " << value << endl;
    cout << "The squared value is " << square << endl;
    return 0;
}
```

وهای مثال من السلايدات
بتعطیه

بتعطیه قيمة وبيعطيك المربع إلها(تربيع)

- C++ provides a function **sqrtf** (square root float) that knows how to compute the square root.
 - This function takes the argument of the *float* type.
 - The result is also a *float* (The root of any number is not always an integer, like the square root of 2).
- to use this function you need to include a header file named **cmath**.



في فنكشن جاهز بلغة الـ C++ (رج)
نعرف شو يعني فنكشن بعدين)
اسمه **sqrtf** بتعطیه قيمة نوعها
float (شوف السنتماكس يلي
بالصورة واحدظهه وبنشرحه بتشابتر
الفنكشنز) وبيعطيك الجذر
التربيعي إلها وبكون الناتج برضو
float (استخدمنا فنكشن عشان
يحسبلنا الجذر كونه م في
operator للجذر في الـ C++)

ولحتى يقدر الكومبايلر يتعرف ع هاذ الفنكشن لازم تكون كاتب

#include <cmath>

مثال لبرنامج بتدخله قيمة ويعطيك الجذر التربيعي لها

```
#include <iostream>
#include <cmath>

using namespace std;

int main(void) {
    float value,squareRoot;

    cout << "Give me a number and I will find its square root:" << endl;
    cin >> value;
    if(value >= 0.0) {
        squareRoot = sqrtf(value);
        cout << "You have given: " << value << endl;
        cout << "The square root is: " << squareRoot << endl;
    }
    return 0;
}
```

في فكرة أخيرة بالتشابير (غالباً ما تكون مطلوبة)

وهي طريقة عرض الأرقام الـ floating point على الشاشة

الأولى اسمها fixed وبتعرض الرقم بالصيغة العاديّة وانت بتعدد كم خانة بعده الفاصلة ، وإذا م بدلت غالباً بعتبرهن 6.

والثانية اسمها scientific وبتعرضه بالصيغة العلميّة وبرضو بتقدر تعدد كم خانة بعده الفاصلة واداً م بدلت بعتبرهم 6 غالباً.

مثال نشوف فيه السنتاكس لكل وحدة :

```
#include <iostream>
using namespace std;
int main()
{
    // Initializing floating point variable
    double a = 4.223234232;
    double b = 2323.0;
    // Printing normal values
    cout << "Normal values of floating point numbers\na = ";
    cout << a << "\nb = " << b << '\n' ;
    // Printing values using fixed
    cout << "Values using fixed \n" ;
    cout << fixed<< a << "\n" << b << '\n' ;
    // Printing values using scientific
    cout << "Values using scientific are : " << endl;
    cout << scientific << a << '\n' << b << '\n' ;
    return 0;
}
```

```
Normal values of floating point numbers
a = 4.22323
b = 2323
Values using fixed
4.223234
2323.000000
Values using scientific are :
4.223234e+00
2.323000e+03
[]
```