

# Chapter 2

## Elementary Programming

Section 2.1-2.13,2.15,2.16

# Variables

ال **variables** : هي عبارة عن أسماء لمتغيرات بحتفظ بداخلهم على **data value** ويتم تخزينها بداخل ال **memory** . ومن الممكن التعديل على هذه القيم خلال البرنامج

- حتى نفهم الفكرة اكثر خلينا نتخيل انه عنا كاسة فاضية :

ممکن اعيبيها مي وتصير كاسة مي

وممكن اعيبيها شاي وتصير كاسة شاي

وممكن اعيبيها طحين وتصير كاسة طحين

فالكاسة هي عبارة عن شيء لاحتواء أي نوع من المواد

- نفس الاشئ ال **variables**، من الممكن انه يحتوي على أي نوع من البيانات

لو كان عنا متغير اسمه **x** وعملنا التالي:

```
x = 1;           // x is 1
x = 2.5;        // x is 2.5
x = 'A';       // x is A
x = "Hi";      // x is Hi
```

لو نلاحظ انه ال **x** في كل مرة احتوى نوع مختلف من ال **data**.

لكن في ال **c++** لازم نحدد شو نوع ال **data** الي رح يحتويها هذا ال **variable**

فرح نستخدم ال **data type** الي وظيفته يحدد نوع محتويات ال **variables**

\*من الممكن انشاء عدد لانهائي من ال **variables** طالما ما بشتركوا بنفس الاسم

# Data Type

- الـ **Data Type** هم المسؤولين عن تحديد نوع الـ **variable** وانواعهم وبتنحط قبل اسم المتغير :

## 1) int

و يتم استخدامها لتخزين الاعداد الصحيحة ، مثال :

```
int x = 1;
```

## 2) float or double

و يتم استخدامها لتخزين الاعداد العشرية - ورح نعرف الفرق بينهم لقدام - ، مثال :

```
double x = 13.5;    float y = 2.135;
```

## 3) char

يتم استخدامها لتخزين الاحرف ، ولازم الحرف يكون بين ' ' ، مثال :

```
char x = ' a ';
```

## 4) string

يتم استخدامها لتخزين النصوص او الجمل لازم يكون النص بين " " ، مثال :

```
string x = " Hi " ;
```

## 4) bool

يتم استخدامها لتخزين القيم المنطقية true او false ( 1 او 0 ) ، مثال :

```
bool x = true ;    bool x = false ;
```

و رح يكون في عنا انواع اخرى من الـ **data type** رح نتعرف عليهم لقدام .

# Identifiers (variables name)

الـ **identifiers** هي طريقة المسموحة لتسمية الـ **variables** والـ **function** هي عبارة عن تسلسل من الاحرف او الارقام او الرموز وتخضع لعدة شروط :

- يجب ان تبدأ بحرف او ( \_ ) او (\$) لكن غير مسموح ان تبدأ **برقم** .
- غير مسموح استخدام رموز الا ( \_ ) وال (\$) .
- غير مسموح انه يكون في مسافة بين الكلمات .
- غير مسموح ان يتم تسميتها باسمااء محجوزة للغة
- غير مسموح بتكرار الاسماء

- من الاسماء المحجوزة باللغة التي تعرفنا عليها :

`include \ iostream \ using \ namespace \ main \ cout \ endl \  
int \ double \ float \ string \ char \ bool \ return`

لغة الـ **C++** تعتبر **sensitive case** يعني انها بتفرق بين الاحرف الصغيرة والكبيرة

$( X \neq x )$

بمعنى:

**Cout** صحيحة وتختلف عن الكلمة المحجوزة **cout**

**name\_space** صحيحة وتختلف عن الكلمة المحجوزة **namespace**

**endl1** صحيحة وتختلف عن الكلمة المحجوزة **endl**

**Example:** which of the following identifiers are valid?



Miles, Test, a++, --a, 4#R, \$4, #44, apps, return1

main , Double , int , x , y , radius , using , void , int

## Declaring Variables & Assignment Statements

**Declaring** : هي انشاء variable جديد وحجز مكانه داخل الـ memory

**Assignment** : هي اسناد او إعطاء قيمة للـ variable

`int x;`            Declaration  
`x = 1;`            Assignment or initialization

ويوجد عدة طرق من الـ **declaration** والـ **initialization** :

(1) على أسطر منفصلة :

```
int x; // declare x to be an int
x = 10; // assignment 10 to x
```

(2) انشاء أكثر من متغير :

```
int x, y; // declare x and y to be int
x = 10; // assignment 10 to x
y = 11; // assignment 11 to y
```

(3) انشاء واسناد قيمة المتغير على نفس السطر :

```
int x = 10, y = 11; // declare & initialization
or
int x(10), y(11);
```

(4) اسناد نفس القيمة للمتغيرات :

```
int x, y, z; // declare x & y & z
x = y = z = 10; // assignment 10 to x & y & z
```

(5) اسناد قيمة متغير الى متغير اخر (نسخ) :

```
int x = 10, y;
y = x; // assignment (x) to y , y = 10
```

**Ex1:** write a program that computes the area of the circle and print it. Reminded: the area of circle is equal ( $radius^2 * \pi$ )

```
#include <iostream>
using namespace std;
int main()
{
    //step1: declare a radius and area as a double
    double radius, area;
    //step2 :assignment the value of radius
    radius = 1;
    //step3: calculate the area
    area = radius * radius * 3.14;
    //step4: print the area
    cout << "the area is = " << area << endl;
}
```

انشاء متغيرين من نوع double

اسناد قيمة 1 الى المتغير radius

اسناد ناتج العملية الى المتغير area  
area=3.14

the area is = 3.14

**خطوة 1:** انشاء المتغيرين **radius & area** في ال memory

**خطوة 2:** اسناد 1 الى المتغير **radius**

**خطوة 3:** اسناد ناتج العملية **radius \* radius \* 3.14** الى المتغير **area**

**خطوة 4:** طباعة **the area is =** وقيمة المتغير **area**

- من الممكن اختصار خطوة 1 و 2 بنفس الخطوة :

```
double radius = 1, area;
```

## Reading Input from the Keyboard or user

- بإمكاننا نسند قيم لل **variables** عن طريق الكيبورد او ال **user** باستخدام ال **(cin)** وهي **function** موجودة في مكتبة ال **iostream** ، ورح نتعلم مع بعض كيف نستخدمها

```
cin >> اسم المتغير ;
```

- ومن الممكن اخذ قيمة لأكثر من متغير في نفس الوقت :

```
cin >> اسم المتغير 1 >> اسم المتغير 2 >> .... ;
```

يتم ادخال القيم على الترتيب ، المتغير 1 ثم المتغير 2 الخ..

- ويفضل قبل كل **cin** نخط جملة **cout** بتوضح شو ال **variable** الي لازم ندخله :

مثال: لو عنا برنامج بياخذ اسم الطالب ورقمه الجامعي لازم يكون كالتالي :

```
string name;  
int number;  
cout << "Enter your name and number:"  
cin >> name >> number;
```

The output :

Enter your name and number :

Ayman

1234

لما نكبس enter بتتخزن القيمة الأولى  
وباخذ القيمة الي بعدها

name = Ayman

number = 1234

**Ex1:** write a program that computes the area of the circle and print it , and take the radius from the user.

```
#include <iostream>
using namespace std;
int main()
{
    //step1: declare a radius and area
    double radius, area;

    //step2: assign the value of radius
    cout << "please enter the radius: " << endl;
    cin >> radius;

    //step3: calculate the area
    area = radius * radius * 3.14;

    //step4: print the area
    cout << "the area is = " << area << endl;
}
```

هون بنعرف انه لازم ندخل  
قيمة ال-radius

```
Please enter the radius : 2
the area is = 12.56
```

**Ex2:** write a program that calculate the avg of 3 numbers reading from user and output the avg. (avg is the sum divided by their number)

```
#include <iostream>
using namespace std;
int main()
{
    //prompt the user to enter three numbers
    double number1, number2, number3;
    cout << "Enter three numbers: ";
    cin >> number1 >> number2 >> number3;

    //compute average
    double average = (number1 + number2 + number3)/3;

    //Display result
    cout << "The average of " << number1 << ", " << number2
        << ", " << number3 << ", is " << average << endl;
}
```

```
Enter three numbers: 20 30 40
The average of 20,30,40 is 30
```



# Named Constants

زي ما بنعرف انه في عنا رموز بالرياضيات الها قيمة ثابتة ما بتتغير ومثال على ذلك ال  $\pi$  ولو احتجنا انه ننشئ variable داخل اللغة اله قيمة ثابتة ما بتتغير بنستخدم ال `(const)`

```
const dataType constVarName = value;
```

منذ لحظة انشاء ال `const variable` القيمة الي اعطينا يها ما بنقدر نغيرها  
ولو حاولنا نعدل على القيمة رح يعطينا `error`

- في امثلة حساب مساحة الدائرة السابقة، من الممكن إضافة متغير `PI` ليحبر عن قيمة الثابت  $\pi$  وبنعطي ال `const` حتى ما يتعدل قيمتها كما يلي :

```
#include <iostream>
using namespace std;
int main()
{
    const double PI = 3.14159;

    // Step 1: Read in radius
    double radius;
    cout << "Enter a radius: ";
    cin >> radius;

    // Step 2: Compute area
    double area = radius * radius * PI;

    // Step 3: Display the area
    cout << "The area is ";
    cout << area << endl;

    return 0;
}
```

انشئنا المتغير `PI`  
كـ `const`

# Numerical Data Types

كما ذكرنا سابقاً أنواع ال Data Type ولكل Type حجم من ال byte الي بتخزن في ال memory الي بحدد اكبر واصغر قيمة ممكن اخزنهم في ال Data type :

int → 4 byte

float → 4 byte

double → 8 byte

char → 2 byte

لكن من الممكن تغيير حجم ال Data type كما يلي :

**بالنسبة للمتغير int**

**Signed integers:**

- **short int** : 2 byte
- **int** : 4 byte
- **long long int** : 8 byte

**Unsigned integers:**

- **unsigned short**: 16 byte
- **unsigned int**: 32 byte
- **unsigned long long**: 64 byte

ومن الممكن اختصار كتابة int بجانب كل وحدة :

**is same as**

**Short int x**

≡

**Short x**

**unsigned short int x**

≡

**unsigned short x**

**unsigned int x**

≡

**unsigned x**

**long int x**

≡

**long x**

**unsigned long int x**

≡

**unsigned long x**

## بالنسبة للمتغيرات الكسرية

### • Floating-point numbers

- **float** : 4 byte يتكون من 7 خانات عشرية
- **double** : 8 byte يتكون من 16 خانة عشرية
- **long double** : 10 byte

```
double x = 10.0, y = 3.0;  
cout << x / y << endl;
```

```
3.3333333333333331 16 digits
```

```
float x = 10.0, y = 3.0;  
cout << x / y << endl;
```

```
3.3333333 7 digits
```

- وحتى اعرف كل Data Type من كم Byte بتكون بنستخدم ال sizeof :

**sizeof (variable or data type)**

```
cout << sizeof(int) << endl;  
cout << sizeof(long long) << endl;  
cout << sizeof(double) << endl;  
cout << sizeof(char) << endl
```

```
4  
8  
8  
1
```

## Binary octal and hex literals

انظمة العد المتعارف عليها هي العشرية (decimal) وهي الي بنستخدمها، أنظمة أخرى

- الثنائي (binary) - والثماني (octal) - والسادس عشر (hexa decimal)

وبنقدر نخزن القيم حسب النظام العد الي بدنا ياه كما يلي:

- التخزين بالنظام الثنائي (binary) :

نضع **0b** او **0B** قبل الرقم ثم نضع الرقم بالنظام الثنائي:

`int x = 0b11010 ;`

رقم بالنظام الثنائي  
ويساوي 26 بالعشري

← تعني ان النظام  
ثماني

- التخزين بالنظام الثماني (octal) :

نقوم بوضع **0** قبل الرقم ثم نضع الرقم بالنظام الثماني:

`int x = 076 ;`

رقم بالنظام الثماني  
ويساوي 62 بالعشري

← تعني ان النظام  
ثماني

- التخزين بالنظام السادس عشر (Hexa decimal) :

نقوم بوضع **0x** او **0X** قبل الرقم ثم نضع الرقم بالنظام السادس عشر :

`int x = 0xA3 ;`

رقم بالنظام السادس عشر  
ويساوي 163 بالعشري

← تعني ان النظام  
السادس عشر

- الـ digits التي تتكون منها كل نظام :

binary : **0 1**

octal : **0 1 2 3 4 5 6 7**

hexa decimal : **0 1 2 3 4 5 6 7 8 9 A B C D E F**

فقط على سبيل المعرفة وليست للحفظ

# remainder operator

- من اهم العمليات الي رح نستخدمها في البرمجة هي باقي القسمة (%)

$$a \% b = n$$

**a** : المقسوم **b** : المقسوم عليه **n** : باقي القسمة

- بعض المعلومات المهمة في لعملية باقي القسمة :

- (1) اذا كان  $(a \geq b)$  يكون جواب باقي القسمة بين الـ  $(0$  و  $b - 1)$
- (2) اذا كان  $(a < b)$  يكون جواب باقي القسمة يساوي  $a$
- (3) لا يمكن ان يكون  $a$  او  $b$  كسور
- (4) إشارة الناتج تكون حسب إشارة الأكبر

- بعض العمليات المهمة :

(1) اذا بدك تجيب اخر خانة من الرقم : **number % 10**

`cout<< 1234 % 10 ;` → 4      `cout<< 1234 % 100 ;` → 34

كل ما بتضيف صفر بتجيب خانة على عدد الصفار

(2) اذا بدك تحذف اخر رقم : **number / 10**

`cout<< 1234 / 10 ;` → 123      `cout<< 1234 / 100 ;` → 12

كل ما بتضيف صفر بتلغي على عدد الصفار

# Exponent Operations

لا يوجد رمز للتعبير عن الأسس ، لكن يوجد طريقة لحساب الأسس عن طريق :

$$\text{pow}(a, b) = (a)^b$$

وهو function موجود في مكتبة ال `cmath` ، رح نتعرف على ال function لقدام

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

لا تنسى إضافة مكتبة ال `cmath` عند استخدامك ل function ال `pow`

```
using namespace std;
int main()
{
```

```
    cout << pow(2.0, 3) << endl;
```

```
    cout << pow(4.0, 0.5) << endl;
```

```
    cout << pow(2.5, 2) << endl;
```

```
    cout << pow(2.5, -2) << endl;
```

```
}
```

8

2

6.25

0.16

# Arithmetic Expressions

كما تعلمنا سابقًا من الممكن استخدام لغة ال C++ لإجراء العمليات الحسابية ، حسب الأولويات الرياضية. ومن الممكن تحويل المعادلات الرياضية الى معادلات بلغة ال C++

Ex: translate this expressions to code:

- المعادلة بشكلها الرياضي :

$$\frac{3 + 4x}{5} - \frac{10(y - 5)(a + b + c)}{x} + 9\left(\frac{4}{x} + \frac{9 + x}{y}\right)$$

$(3 + 4 * x) / 5$	$(10 * (y - 5) * (a + b + c)) / x$	$9 * (4 / x + (9 + x) / y)$
-------------------	------------------------------------	-----------------------------

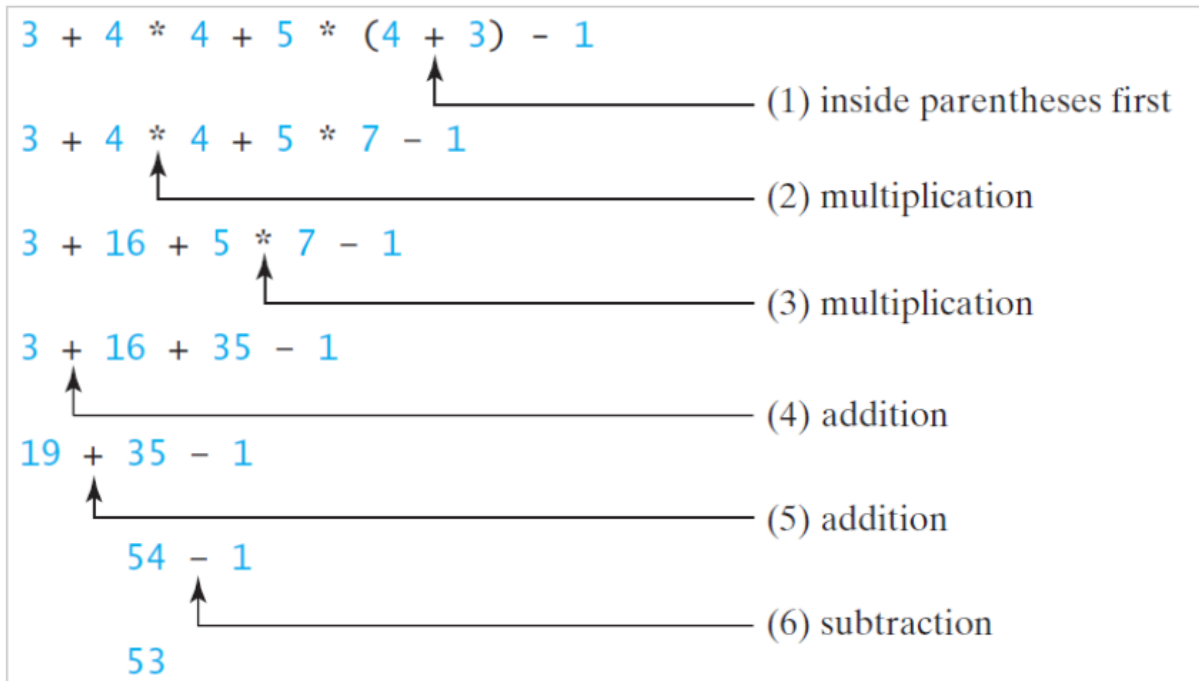
- المعادلة بلغة ال C++ :

$$((3 + 4 * x) / 5) - (10 * (y - 5) * (a + b + c) / x) + (9 * (4 / x + (9 + x) / y))$$

## لتذكير بالأولويات الحسابية :

الاقواس -> باقي القسمة و الضرب و القسمة من اليسار -> الجمع و الطرح من اليسار

### - مثال على الأولويات الحسابية :



ex : Write a program that converts a Fahrenheit degree to Celsius using the formula:  **$Celsius = (\frac{5}{9})(Fahrenheit - 32)$**

```
#include <iostream>
using namespace std;
int main()
{
    double celsius, fahrenheit;

    cout << "Enter the fahrenheit degree: ";
    cin >> fahrenheit;

    celsius = (5.0 / 9) * (fahrenheit - 32);
}
```



## Augmented Assignment Operators

إذا كنا نبدأ نجري عملية حسابية على variable ونخزن الجواب النهائي على نفسه  
بنستخدم ال augmented assignment operators.

- وهي عبارة عن دمج العمليات الحسابية ( \* ، / ، + ، - ، % ) مع المساواة .

Operator	Name	Example	equivalent
+=	Addition assignment	i += 8	i = i + 8
-=	Subtraction assignment	i -= 8	i = i - 8
*=	Multiplication assignment	i *= 8	i = i * 8
/=	Division assignment	i /= 8	i = i / 8
%=	Modulus assignment	i %= 8	i = i % 8

```
int x = 4;
```

```
x = x + 1;
```

قيمة x هي 4

x = 4 + 1 = 5

is same as

```
int x = 4;
```

```
x += 1;
```

لا يوجد مسافة بين  
العملية والمساواة

+= +=

### Examples :

```
int x = 4, y = 2;
```

```
x -= 1;
```

```
cout << x << endl;
```

3

```
x /= y;
```

```
cout << x << endl;
```

2

```
x *= 2+6/3;
```

```
cout << x << endl;
```

16

```
x %= 2 ;
```

```
cout << x << endl;
```

0

```
x *= y;
```

```
cout << x << endl;
```

8

الأولوية للعمليات على يمين المساواة ثم الي على اليسار

2+3/6 = 2+2 = 4 -> x\*4 = 16

# Increment and Decrement Operators

زيادة او النقصان . وهما من العمليات المشهورة وكثيرة الاستخدام

(Increment) هي زيادة قيمة المتغير بمقدار 1 ، عن طريق ++

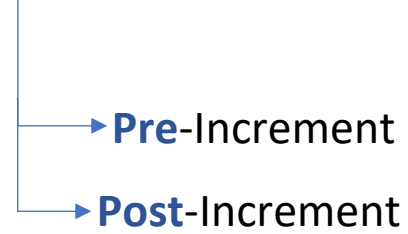
(Decrement) هي نقصان قيمة المتغير بمقدار 1 ، عن طريق --

ويوجد نوعين من العمليات :

## Decrement



## Increment



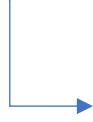
Operator	Name	Description	Example (i=1)
<b>++var</b>	pre-increment	بزيد على ال var قيمة 1 واستخدام القيمة المعدلة	<pre>int j = ++i // j is 2, i is 2</pre>
<b>var++</b>	Post -increment	نستخدم قيمة ال var القديمة بعدين نزيد عليها 1	<pre>int j = i++ // j is 1, i is 2</pre>
<b>--var</b>	pre-decrement	بنقص على ال var قيمة 1 واستخدام القيمة المعدلة	<pre>int j = --i // j is 0, i is 0</pre>
<b>var--</b>	post-decrement	نستخدم قيمة ال var القديمة بعدين بنقص منها 1	<pre>int j = i-- // j is 1, i is 0</pre>

دائما تتغير قيمة i لكن الفكرة ، قيمة j تعتمد على قيمة i قبل او بعد التعديل

- مثال توضيحي للتمييز بين ال pre وال post :

```
int x = 1 ;
```

```
int y = ++x ;
```



اول اشي زيد على قيمة x واحد



ثم خزن القيمة بعد التعديل في y

```
X = 2  
Y = 2
```



```
int x = 1 ;
```

```
int y = x++ ;
```



اول اشي خزن قيمة x القديمة في y

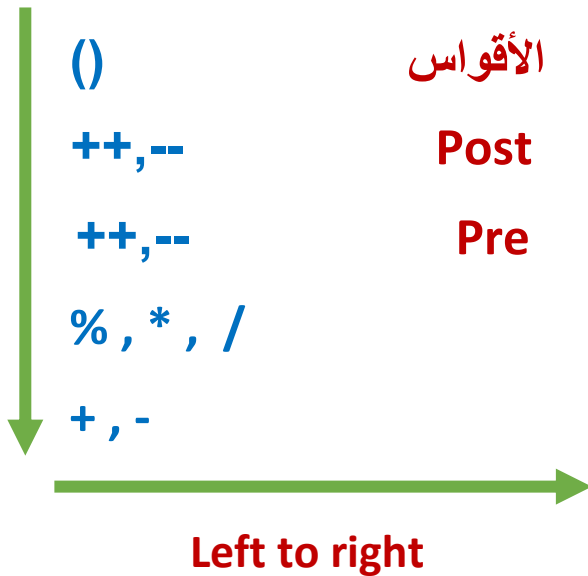


ثم ضيف على قيمة x واحد

```
X = 2  
Y = 1
```



- الأولويات الحسابية مع ال pre وال post :



Examples : find the final value of x and j :

```
int x = 5;  
int j = x++;  
x = 6 , j = 5
```

```
int x = 5;  
int j = ++x;  
x = 6 , j = 6
```

```
int x = 5;  
int j = x--;  
x = 4 , j = 5
```

```
int x = 5;  
int j = --x;  
x = 4 , j = 4
```

```
int x = 5;  
int j = x++ + 10;  
x = 6 , j = 15
```

```
int x = 5;  
int j = x++ + ++x;  
x = 7 , j = 12
```

```
int x = 5;  
int j = x++ + x;  
x = 6 , j = 10
```

```
int x = 5;  
int j = x + --x;  
x = 4 , j = 8
```

# Type conversion

هو عبارة عن أمر يقوم بتحويل ال Data type لل variable الى واحد اخر

**static\_cast** <type> (value or variable)

ال type الي بدى احوال عليه

المتغير او القيمة الي بدى احوالها

is same as

**(type) value or variable**

```
double x = 4.3;
```

```
cout << static_cast<int>(x); same as cout << (int)x;
```

فيما سبق، ال casting ما بغير من قيمة ال- x الاصلية ورح اتضل زي ما هي 4.3

```
double x = 2.5;
```

```
x = static_cast<int>(x);
```

هون عدلنا على نفس قيمة المتغير x واصبح int وقيمتة تساوي 2

```
double x = 2.5, y;
```

```
y = static_cast<int>(x);
```

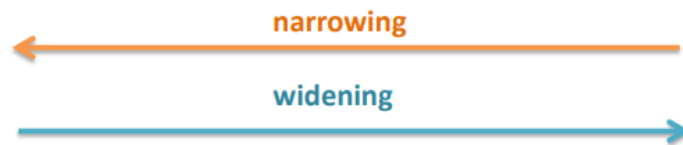
هون عدلنا على قيمة y واصبحت 2 من نوع int ، لكن ال- x ضل زي ما هو

- يوجد عدة انواع من التحويل، ان يكون واحدة من التالي :

**Narrowing:** التضييق ، كلما اتجهنا لليساار يضيق مجال المتغير

**Widening:** التوسيع ، كلما اتجهنا لليمين يتوسع مجال المتغير

short	int	long long	float	Double
-------	-----	-----------	-------	--------



```
double x = 2.5;  
x = static_cast<int>(x); // double to int (narrowing)
```

```
int x = 2;  
cout << static_cast<double>(x); // int to double (widening)
```

- وان يكون واحدة من التالي :

**Implicit :** ضمني ، وهو ان تتم عملية التحويل تلقائيًا من غير كتابتها

**Explicit :** صريح ، هو ان اكتب امر التحويل في الكود

```
int x = 2.5; //implicit سيتم تجاهل ال 0.5 تلقائيًا وتحويله الى عدد صحيح
```

```
double x = 2.5;  
x = static_cast<int>(x); //explicit تمت عملية التحويل صريحةً
```

- تعرفنا على الـ char سابقاً، ويستخدم لتخزين حرف او رقم او رمز واحد فقط  
ولكل حرف الـ رقم يعبر عنه ، كيف يعني ؟

**ASCII Table**: جدول يمثل القيمة العددية لكل حرف :

Character	ASCII
<b>A</b>	<b>65</b>
<b>B</b>	<b>66</b>
<b>C</b>	<b>67</b>
.	.
.	.
<b>Z</b>	<b>90</b>

Character	ASCII
<b>a</b>	<b>97</b>
<b>b</b>	<b>98</b>
<b>c</b>	<b>99</b>
.	.
.	.
<b>z</b>	<b>122</b>

كما نلاحظ انه كل حرف الـ رقم بمثله خاص في ما يشترك مع أي حرف اخر  
وايضاً نلاحظ التسلسل الرقمي حسب الترتيب الابجدي حيث كل حرف يزيد عن الحرف الي قبله بواحد  
، ويبدأ من الحرف (A) ، (a) وينتهي بالحرف (Z) ، (z).

- وبنسبة للرقم اذا تم وضعه بين ' ' فهو يعتبر حرف وتمثيله على الـ ASCII :

Character	ASCII
<b>0</b>	<b>48</b>
<b>1</b>	<b>49</b>
<b>2</b>	<b>50</b>
.	.
<b>9</b>	<b>57</b>

- Casting with char :

- التحويل من char الى int :

```
int i = static_cast<int>('a');
```

i = 97

قيمة الـ a كرقم = 97

- التحويل من int الى char :

```
char c = static_cast<char>(97);
```

c = 'a'

قيمة الـ 97 كحرف = a

- من الممكن ان يكون التحويل بالطريقة الضمنية :

```
int i = 'a';
```

```
char c = 97;
```

## - Numeric Operators on Characters:

- من الممكن اجراء عمليات حسابية على الـ char :

```
1- int j = 2 + 'a';
```

حسب الـ ASCII فقيمتها تساوي 97

$j = 2 + 97 \rightarrow j = 99$

```
2- int i = '2' + '3';
```

تعتبر char وحسب الـ ASCII فبتحول قيمتها الى 50

تعتبر char وحسب الـ ASCII فبتحول قيمتها الى 51

$i = 50 + 51 \rightarrow i = 101$

```
3- char c = 'b';
```

c++

رح نزيد 1 ، ورح تنتقل الـ b الى الحرف الي بعده

c = 'c'

c--

رح نطرح 1 ، ورح تنتقل الـ b الى الحرف الي قبله

c = 'a'

```
4- char x = 'a' + 2;
```

الـ a تساوي 97 ،  $97 + 2 = 99$

ولان الجواب النهائي char ، فالحرف الي بساوي 99 هو c

x = 'c'



# overflow

كما تعلمنا سابقاً لكل data type سعة تخزين محددة ، وإذا قمنا بتخزين قيمة أكبر من السعة المحددة في هذه الحالة يحدث ال overflow

- وفي هذا الجدول يظهر ال range لبعض الأنواع الأساسية :

Data Type	Range
int	-2147483648 to 2147483647
char	-128 to 127
double	$-1.7 \times 10^{308}$ to $1.7 \times 10^{308}$
float	$-3.4 \times 10^{38}$ to $3.4 \times 10^{38}$

ليست للحفظ

وعلينا تجنب حدوث ال overflow للحفاظ على قيم ال variable الصحيحة

- ويحدث ال overflow في عدة حالات منها :

حاصل جمع رقمين أكبر من ال range :

```
int x = 2147483647 + 1;
```

حاصل طرح رقمين أصغر من ال range :

```
int x = -2147483647 - 1;
```

## Problems with answers:

- fill the missing parts in the following questions:

1) Create a variable named (**number**) and assign the value (**50**) to it.

```
  =  ;
```

2) Display the sum of (5+10), using two variables: x and y .

```
  =  ;  
int y = 10 ;  

```

3) Create a variable called z, assign x + y to it, and display the result.

```
int x = 5 ;  
int y = 10 ;  
  = x + y ;  

```

4) fill in the missing parts to create three variables of the same type using ,

```
 x = 5  y = 6  z = 50 ;  
cout << x + y + z ;
```

5) What are the values stored in variables x, y, and z after executing the following code?

```
int x = 34, y , z;  
y = x++;  
z = ++x;  
x %= 4;
```

x	y	z
<input type="text"/>	<input type="text"/>	<input type="text"/>

6) fill in the missing parts to print the sum of two numbers (by user):

```
int x, y ;  
int sum ;  
cout << "Enter the first number: ";  
 >>  ;  
cout << "Enter the second number:  
 >>  ;  
sum = x + y;  
cout << "sum is : " <<  ;
```

7) add the correct data type for the following variables :

```
 myNum = 9 ;  
 myDoubleNum = 8.99 ;  
 myLetter = 'A' ;  
 myBool = false ;  
 myText = " Hello World " ;
```

8) write the type of casting of following question :

```
int x = 1.2;           Implicit casting with type narrowing  
double n2 = 3.4;  
float n3 = (float)n2; 
```

- write code for the following questions :

1) print number entered by **User** :

Sample test input :                      Ouput:

Enter the number : 1                      The number is 1

2) program to add **two** entered numbers:

Sample test input:                      output:

Enter two numbers : 1 ,2                      The sum of 1 and 2 = 3

3) find the size of **int** , **float** , **double** and **char**

Output :

The size of int is : 4

The size of float is : 4

The size of double is : 8

The size of char is : 1

4) **swap two** variables

Sample test input :                      output:

Enter x and y : 1 2                      The x and y before swap is 1 2

The x and y after swap is 2 1

5) find the average of **4 numbers** :

Sample test input :                    output:

Enter 4 numbers : 2 4 6 8    the average is = 5

6) find the **square** of **N**:

Sample test input:                    output:

Enter the number : 2                    the square of 2 is 4

7) write a program that take number in **feet**, then **convert** it into meters

**1 foot is 0.305 meter**

Sample test input:                    output:

Enter the feet number: 3                    the 3 in meter is = 0.915

8) reads the number between **10** and **99**. And print the sum of the **2 digit**

Sample test input:                    output:

Enter number : 33                    the sum = 6

9) read **3** numbers **x y z**, and find the  **$(x^y)^z$**  :

Sample test input:                    output:

Enter x y z : 2 , 2 , 2                     $x^y^z = 16$

- Answers of the questions :

1)int number = 50;

2)int x = 5;  
cout<<x+y;

3)int z = x+y;  
cout<< z;

4)int x = 5, y =6, z = 50;

5)x = 3 , y =34 , z = 36

6)cin>>x;  
cin>>y;  
cout<<"sum is:"<<sum;

7)int / double / char / bool / string

8)explicit casting with type narrowing

```
1) #include <iostream>
using namespace std;
int main()
{
    int x;
    cout << "Enter the number:";
    cin >> x;
    cout << "The number is " << x;
}
```

```
2) #include <iostream>
using namespace std;
int main()
{
    int x, y;
    cout << "Enter two numbers:";
    cin >> x >> y;
    cout << "The sum of " << x << "and" << y << "=" << x + y;
}
```

```
3) #include <iostream>
using namespace std;
int main()
{
    cout << "The size of int" << sizeof(int) << endl;
    cout << "The size of float" << sizeof(float) << endl;
    cout << "The size of double " << sizeof(double) << endl;
    cout << "The size of char " << sizeof(char) << endl;
}
```

```
4) #include <iostream>
using namespace std;
int main()
{
    int x, y;
    cout << "Enter x and y :";
    cin >> x >> y;
    cout << "The x and y before swap is " << x << " " << y << endl;

    int z;
    z = x;
    x = y;
    y = z;
    cout << "The x and y after swap is " << x << " " << y;
}
```

```

5) #include <iostream>
#include<cmath>
using namespace std;
int main()
{
    double w, x, y, z;
    cout << "Enter 4 numbers";
    cin >> w >> x >> y >> z;
    double avg = (w + x + y + z) / 4;
    cout << "the average is = " << avg;
}

```

```

6) #include <iostream>
using namespace std;
int main()
{
    int N;
    cout << "Enter the number";
    cin >> N;
    cout << "the square of " << N << " is " << N * N;
}

```

```

7) #include <iostream>
using namespace std;
int main()
{
    double x;
    cout << "Enter the feet number:";
    cin >> x;
    cout << "the " << x << " in meter is " << x * 0.305;
}

```

```

8) #include <iostream>
using namespace std;
int main()
{
    int x;
    cout << "Enter number:";

    int sum = x % 10 + x / 10;
    cout << "The sum = " << sum;
}

```

```

9) #include <iostream>
#include<cmath>
using namespace std;
int main()
{
    int x, y, z;
    cout << "Enter x y z";
    cin >> x >> y >> z;
    cout << pow(pow(x, y), z);
}

```