



Fundamentals Of Programming with C++

Fundamentals Of Programming With C++ ملخص كورس



A lot of thanks to [Osama Elzero](#)

Created By: [Fady Alamir](#)



#001 - Important Introduction About The Course

1. Before the journey

- ١- اترك كل شيء في وقته
- ٢- عند تعلم شيء في الكورس ابحث عنه وزود من معلوماتك عنه
- ٣- عندما لا تفهم شيء لا تتركه وتتدخل على اللي بعده عيد الفيديو ولو مفهمنتش بردك اسأل وطبيعي إنك متفهمنش من اول مرة
- ٤- اسمع كلام مدرب الكورس لو قلك اتفرج على فيديو معين اتفرج عليه
- ٥- أنت داخل تتعلم برمجة وليس لغة C++ (ما هي إلا اداة لتعلم البرمجة)
- ٦- خليك شخص محترف أبدع وابتكر ونفذ اي فكرة من دماغك

1. What can you do with C++?

- | | |
|-----------------------|-----------------|
| 1- Gaming | 2- Desktop Apps |
| 3- OS | 4- Web Browsers |
| 5- Servers Management | |

3. Course Content

- | | |
|---|--------------------------------------|
| 1- Fundamentals of Programming with C++ | 2- Object Oriented Programming (OOP) |
| 3- Algorithms | 4- Data Structure |
| 5- Problem Solving | |

4. What you need to start?

- 1- A Reason to start: هدف يخليك تكمل وتواجه الصعوبات
- 2- [Before programming playlist](#)
- 3- Be Patient & Calm
- 4- Every time we will add extensions and tools to help us



#002 - Why C++ Language

1- Widely Used

2- Very Fast

3- Has Pointers

4- Portable “Cross Platform”: يمكن تشغيلها على أكثر من OS

5- Closer to Hardware

6- Strong language + Covers what you need

7- Support procedural programming and object oriented programming

8- Universities

9- References + Resources + Community

10- Only C++?

11- Will I work with C++?

#003 - Install VSC Editor, Compiler And Debugger

Done

#004 - Install Visual Studio And Answer Questions

Done

#005 - How The C++ Works

١- المكتبات Libraries: هي مكتبة جاهزة بها مجموعة من الـ Functions نستدعيها لنسخدم الـ Functions الموجودة بها في الملف.

٢- الـ Source file: هو الملف الذي يقوم بداخله بكتابة الكود.

٣- الـ Source code: هو الكود المكتوب داخل الملف.

٤- الـ Hashtag sign (#): عبارة عن preprocessor statement بمعنى معالجة لفكرة معينة قبل "pre" قبل عملية ترجمة الكود الـ Compile للغة الكمبيوتر.

٥- الـ include directive: هي preprocessor directive تستعمل لتضمين الملف أي مبدأ الـ File inclusion أي نحضر محتوى المكتبة كاملة داخل المشروع الخاص بنا.

٦- الـ iostream: هو عبارة عن file بداخله مجموعة من الـ Functions تحتاج الى استخدام مجموعة من الـ functions بداخله في الملف الخاص بنا.

٧- الـ Function: دالة برمجية بها مجموعة من الخطوات تتم خطوة بخطوة أو خطوة واحدة.



٨- يتم تنفيذ الكود في البرنامج **line by line** بالترتيب وبالتحكم في ال **flow, stream** ستقرر من الذي ي Run الأول.

٩- نوع من أنواع البيانات يسمى **integer** ونوع بياناته أرقام بحيث أن **function** **int** مثل () يقوم بإرجاع رقم **.return(0);**

#006 - Preprocessing, Compiling And Linking

١- الملف **iostream** يقوم باستدعاءه ويسمى **header file** امتداده يكون **.h**. أو امتدادات أخرى ومن الممكن ألا يكون له امتداد.

٢- لغة C++ ليس لها علاقة بالامتداد ولا الملفات الـ **files** عبارة عن **containers** بها مجموعة من الكلمات نعطيها للـ **compiler** حيث أن الكود أو المحتوي هو الأهم.

```
1 #include <iostream>
2
3 int main()
4 {
5     std::cout << "Line 1\n";
6     std::cout << "Line 2\n";
7     std::cout << "Line 3\n";
8     return 0;
9 }
```

٣- إنشاء ملف الـ **Translation Unite** سوف يكون بداخله محتوى الـ **Library** مع البرنامج الخاص بك سوف يأتي الـ **Compile** ويقوم بترجمته للغة الآلة حيث أن هذه الترجمة تتم في **Object file** يكون امتداده **.obj**.

Name	Date modified	Type	Size
Test.tlog	3/13/2023 12:40 AM	File folder	
Source.obj	3/13/2023 12:40 AM	Object File	50 KB
Test.log	3/13/2023 12:40 AM	Text Document	1 KB
vc142.idb	3/13/2023 12:40 AM	VC++ Minimum R...	147 KB
vc142.pdb	3/13/2023 12:40 AM	Program Debug D...	396 KB

الـ **object file** هو الذي يستطيع الكمبيوتر فهمه وهو ليس برنامج يمكن تشغيله هو ملف ينتظر الـ **Linker** منك حيث يقوم بـ **link** كل الـ **objects** ليخرج الـ **exe file** النهائي الذي تقوم بتشغيل البرنامج من خلاله.



#007 - C++ Language Syntax

- ١- الـ **Syntax**: مجموعة القواعد التي تحكم استخدام الأسطر البرمجية وبنية اللغة لينتج ما تريده اللغة بالضبط ولا يحدث أي **error**.
- ٢- الـ **Curly Braces** { } : هما الـ **Block of containers** أي الحاوية التي تحتوي على **code** صندوق الأكواد.
- ٣- كل سطر برمجي ينتهي بـ ; أي **semicolon**
- ٤- هذه العلامة : تسمى **colon** وعندما تكون :: تسمى **double colon**
- ٥- اختصار **cout** : **character output**
- ٦- **Output** : المخرجات **Input** - المدخلات
- ٧- كل حرف او علامة يُسمى **character** في الـ **space** والـ **code** المسافة الفارغة تسمى **character**
- ٨- الـ **operator ← <<** حيث أن هذا الـ **stream insertion operator** مسؤول عن الادخال يرسل البيانات التي تليه للـ **function** لخروج للـ **stream**
- ٩- **Syntax highlighter**: يقوم بتمييز كل كود بلونه المميز.
- ١٠- نوع من البيانات يسمى **string** ويضاف داخل الـ **double quotes**
- ١١- أي شيء يأتي بعد الـ \ \ أي **special character end backslash** يسمى **new line** مسؤولة على أن يقوم بعمل شيء مميز مثل \n
- ١٢- لا يهم الـ **compiler** كيف الكود مكتوب لكن له علاقة بالسطر البرمجي ما شكله وما المكتوب به.

#008 - Comments And Use Cases

- ١- الـ **comment** : عبارة عن **Note** أو وصف للكود الذي قمت بكتابته، ويكتب الـ **// - double forward slash comment**

single line comment يسمى `// Search Google For "Iostream"` **comment** وهذا الـ

Multiple line comment - ٢

```
/*
=====
|   Falcon
=====
*/
```

ويتمكن استخدامه كـ **single line comment**

```
std::cout /* <= This Is Character Out */ << "Line 2\n";
```



٣- الـ Comment لا يستخدم لوصف الكود فقط ولكن يستخدم لمنع السطر من الـ compile عندما تقوم بعمل run للكود

٤- من / يمكن عمل comment or uncomment للسطر أو لعدة سطرين بعد تحديدهم

#009 - Variables Basic Knowledge

١- المتغير: عبارة عن Data container حاوية للبيانات بحيث تخزن هذه البيانات في الذاكرة Memory

٢- أي شيء داخل الـ rapped in double quotes أي Double Quotes كما هي بدون تطبيق العمليات الحسابية أو أي شيء آخر هكذا

```
std::cout << "\nPrice After Adding 15 Is: 100 + 15";
```

```
Fundamentals Of Programming With C++\#009 - Variables Basic Knowledge\app2
Price Is: 100
Price After Adding 15 Is: 100 + 15
```

٣- إنشاء متغير أو التصريح عن وجود متغير جديد أو الإعلان عن وجود متغير جديد.

Syntax

```
- [Data_Type] [Variable_Name] = [Value]
```

٤- نوع المتغير int: أي رقم وهو عدد صحيح فقط.

٥- يمكن عمل update لـ value المتغير هكذا

```
int main()
{
    int price = 200;
    std::cout << "Price Is: " << price;
    std::cout << "\nPrice After Adding 15 Is: " << price + 15 ;
    std::cout << "\nPrice After Adding 50 Is: " << price + 50 ;
    price = 150; ←
    std::cout << "\nThe New Price Is: " << price;
    return 0;
}
```

٦- يمكن ألا نقوم بكتابة الـ namespace كل مرة وهو الـ std:: هكذا

```
#include <iostream>
using namespace std; ←

int main()
{
    int price = 200;
    cout << "Price Is: " << price;
    cout << "\nPrice After Adding 15 Is: " << price + 15 ;
    cout << "\nPrice After Adding 50 Is: " << price + 50 ;
    price = 150;
    cout << "\nThe New Price Is: " << price;
    return 0;
}
```



#010 - Variables Naming Rules And Best Practices

١ - **Naming Rules**: عندما نقوم بإنشاء متغير جديد يجب أن نتبع قوانين معينة في اختيار اسم المتغير.

٢ - **Best Practices**: أحسن الممارسات التي يقوم باستخدامها الآخرين عند اختيار اسم المتغير.

- Naming Rules
 - Must Be Unique
 - Case Sensitive
 - Cannot Start With Numbers
 - Nums Or Letters Or Underscore
 - No White Space Or Special Characters
 - Reserved Keywords "Class, Public"

- Best Practices
 - Related Names
 - Writing Style

١ - **Must Be Unique**: بحيث لا يمكن تسمية متغيرين بنفس الاسم في الكود.

٢ - **Case Sensitive**: الحروف حساسة في الكود مثل عندما نقوم بتسمية متغير `price` ومتغير آخر `error` لا يحدث `Price` لأنهم مختلفين

٣ - **Cannot Start With Numbers**: لا يمكن بدء اسم المتغير بـ رقم

```
app.cpp:27:9: error: expected unqualified-id before numeric constant
27 |     int 1num = 10;
      |           ^~~~
```

٤ - **Nums Or Letters Or Underscore**: يمكن وضع الأرقام والحروف الإنجليزية والـ `.errors` في المنتصف أو في آخر اسم الـ `variable` بدون وجود `underscore`

وـ `underscores` يمكن وضعها في بداية اسم الـ `variable` بدون `error` أيضًا.

```
30 |     int _numbers = 100;
31 |     cout << _numbers;
32 |     return 0;
33 | }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
100



٥. **No White Space Or Special Characters** في اسم white space : عند استعمال أي Identifier يعتروا كلمتين لذا يحدث error .

```
30 |     int _num_ bers_ = 100;
31 |     cout << _num_bers_;
32 |     return 0;
33 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

School\#2 - Fundamentals Of Programming With C++\#010 - Variables Naming Rules And Best Practices\app.cpp: In function 'int main()':
app.cpp:30:15: error: expected initializer before 'bers_'
30 | int _num_ bers_ = 100;
| ^~~~~~

٦. **الـ special character** في أي مكان في اسم المتغير ممنوعة

```
30 |     int _num_@bers_@ = 100;
31 |     cout << _num_bers_;
32 |     return 0;
33 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Variables Naming Rules And Best Practices\School\#2 - Fundamentals Of Programming With C++\#010 - Variables Naming Rules And Best Practices\app.cpp:30:14: error: stray '@' in program
30 | int _num_@bers_@ = 100;
| ^
app.cpp:30:20: error: stray '@' in program
30 | int _num_@bers_@ = 100;
| ^

٧. **يوجد كلمات محجوزة في اللغة لا يمكن استخدامها في تسمية الـ variable** : Reserved Keywords “Class, Public” .

C++ Reserved Words

asm	double	new	switch
auto	else	operator	template
break	enum	private	this
case	extern	protected	throw
catch	float	public	try
char	for	register	typedef
class	friend	return	union
const	goto	short	unsigned
continue	if	signed	virtual
default	inline	sizeof	void
delete	int	static	volatile
do	long	struct	while



وعند تسمية الـ **variable** بأي اسم من هذه الأسماء يحدث **error**

```

33     int public = 1000;
34     cout << public;
35     return 0;
36 }

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
app.cpp:33:9: error: expected unqualified-id before 'public'
33 |     int public = 1000;
|           ^~~~~~
```

:Writing Styles .٧

Example	Name	Languages
elzeroWebSchool	Camel Case	C, C++, JavaScript
ElzeroWebSchool	Pascal Case	Python
elzero_web_school	Snake Case	
elzero-web-school	Kebab Case	
ELZERO_WEB SCHOOL	Screaming Snake Case	C#

أسم معبر عن المتغير يكون مفهوم .٨:Related Names

#011 - Variables Advanced Knowledge

Variables Advanced Knowledge

- Declare Variable Without Value + Change Later
- Declare Multiaple Variables Without Value + Change Later
- Decalre Multiaple Variables With Same Value

١- يمكن الإعلان عن المتغير فقط بدون إعطائه قيمة.

٢- يمكن الإعلان عن أكثر من متغير معاً

```

int b, c, d;
b = 10, c = 20, d = 30;
cout << b + c + d; // 60 => [10 + 20 + 30]
```



٣- يمكن الإعلان عن عدة متغيرات واعطائهم نفس القيمة

```
int h, i, j;
// h = 10, i = 10, j = 10;
h = i = j = 10;
cout << h + i + j;
```

#012 - Variables Scope

Variables Scope

- Global Variable
- Local Variable

functions هي التي يمكن استدعاءها من قبل جميع الـ Global Variable - ١

```
10 int a = 100; // Global Variable
11 int b = 200; // Global Variable
12
13 int second()
14 {
15     cout << a << " Coming From Second Function\n";
16     cout << b << " Coming From Second Function\n";
17     return 0;
18 }
19
20 int main()
21 {
22     cout << a << " Coming From Main Function\n";
23     cout << b << " Coming From Main Function\n";
24     second();
25     return 0;
26 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Variables Scope\" && g++ app.cpp -o app && "f:\Programming Programming With C++\#012 - Variables Scope\"app
100 Coming From Main Function
200 Coming From Main Function
100 Coming From Second Function
200 Coming From Second Function
```



هي التي توجد داخل function واحدة معينة ولا يستطيع أي آخر أن ي'accsess' عليه Local Variable - ٢

```

10 int a = 100; // Global Variable
11
12 int second()
13 {
14     cout << a << " Coming From Second Function\n";
15     cout << b << " Coming From Second Function\n";
16     return 0;
17 }
18
19 int main()
20 {
21     int b = 200; // Local Variable
22     cout << a << " Coming From Main Function\n";
23     cout << b << " Coming From Main Function\n";
24     second();
25     return 0;
26 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
app.cpp: In function 'int second()':
app.cpp:15:13: error: 'b' was not declared in this scope
  15 |     cout << b << " Coming From Second Function\n";
     |           ^

```

```

10 int a = 100; // Global Variable
11
12 int second()
13 {
14     int b = 200; // Local Variable
15     cout << a << " Coming From Second Function\n";
16     cout << b << " Coming From Second Function\n";
17     return 0;
18 }
19
20 int main()
21 {
22     cout << a << " Coming From Main Function\n";
23     cout << b << " Coming From Main Function\n";
24     second();
25     return 0;
26 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
app.cpp: In function 'int main()':
app.cpp:23:13: error: 'b' was not declared in this scope
  23 |     cout << b << " Coming From Main Function\n";
     |           ^

```

#013 – Constant Variable

Constant Variable

- Read Only Value
- Can't Declare Without Value

هو متغير ثابت للقراءة فقط لا يمكن تغييره أو التعديل عليه.

```

10 int main()
11 {
12     const int num = 100;
13     num = 200;
14     cout << num;
15     return 0;
16 }
```

app.cpp: In function 'int main()':
app.cpp:13:9: error: assignment of read-only variable 'num'
 13 | num = 200;
 | ^~~~~~

٢- لا يمكن الإعلان عن متغير ثابت دون قيمة لأنه لا يمكن التعديل على هذه القيمة مرة أخرى.

```
app.cpp: In function 'int main()':
app.cpp:16:15: error: uninitialized 'const x' [-fpermissive]
  16 |     const int x;
     |           ^

```

٣- يوجد طريقة أخرى لإنشاء متغير ثابت منها بواسطة preprocessor directive

`#define DAYS 9`

```

19     cout << "\n"
20     | << DAYS;
21     return 0;
22 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
[Running] cd "f:\Programs\Variables - Constant Variables\Fundamentals Of Programming"
100
9
```



#014 – Calculate Your Age Application

تستخدم cout للإعلان عن المخرجات من البرنامج لكن cin تعبر عن المدخلات

```
int age;  
cin >> age;
```

#015 - Escape Sequences Characters

Escape Sequences Characters

- --- Special Non Printing Characters
- --- Control Printing Behaviour
- --- Start With Back Slash "\ "
- --- Can Be Inserted In Any Position
- \n
- \\
- \"
- \'
- \t => Tab Equal 8 Spaces
- \b
- \a => Alert => Play Beep During Execution
- \r => Carriage Return

يوجد بعض الحروف المميزة التي من خلالها نستطيع أن نتحكم في السلوك الخاص بالطباعة.

أحرف مميزة لا تطبع مثل \n : Special Non Printing Characters - ١

نتحكم في سلوك الطباعة : Control Printing Behavior - ٢

جميع الـ characters تبدأ بـ \ : Start with Back Slash "\ " - ٣
الـ characters اللي بعده.

The screenshot shows a code editor with the following code in the main() function:

```
main()  
{  
    cout << "missing terminating " character gcc  
    cout << "missing terminating " character gcc  
    cout << "View Problem (Alt+F8) No quick fixes available  
    cout << "Line 3\"|
```

A tooltip from the IDE indicates a syntax error: "missing terminating " character gcc". A "View Problem (Alt+F8)" link and "No quick fixes available" message are also visible.

يمكن استعمالها في أي مكان في الكود. - ٤



٥- لطباعة \ في ال stream نستخدم \\ حيث يقوم بعمل

skip للذى بعده.

```

20 int main()
21 {
22     cout << "Line 1\n";
23     cout << "Line 2\n";
24     cout << "Line 3\\";
25     return 0;
26 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[Running] cd "f:\Programming\4- Elzero Web Sequences Characters" && g++ app.cpp -o a Programming With C++#015 - Escape Sequenc Line 1 Line 2 Line 3\\ [Done] exited with code=0 in 1.99 seconds

٦- لطباعة \ في ال stream لا يمكن اضافتها هكذا

```

25     cout << "Line \"4\"";
26     return 0;
27 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Programming With C++#015 - Escape Sequences Characters\app app.cpp: In function 'int main()': app.cpp:25:20: error: expected ';' before numeric constant 25 | cout << "Line \"4\"";
| ^
| ;

لأن ال compiler تعامل مع ال double quotes برمجياً وليس كأنها مكتوبة للطباعة بحيث أن Line 4 قبلها start quote وبعدها closing quote وخارجهم لذا يحدث error ويحتاج إلى ; يمكن حل هذه المشكلة هكذا بوضع back slash قبل كل من ال start quote وال closing quote

```

20 int main()
21 {
22     cout << "Line 1\n";
23     cout << "Line 2\n";
24     cout << "Line 3\\\\n";
25     cout << "Line \'4\\\\\\n";
26     return 0;
27 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Line 1
Line 2
Line 3\\
Line "4"

٧- تستخدم ال single quote back slash لـ أيضاً.

٨- ال tab وهي عبارة عن طريقة 8 white space وتنسيق \t وتسمي special IDE أو ال text editor ويمكن تغيير ال spaces من ال skipping character

```

int main()
{
    cout << "Line 1\\n";
    cout << "Line 2\\n";
    cout << "Line 3\\\\\\n";
    cout << "Line \'4\\\\\\n";
    cout << "Line \'5\\\\\\n";
    cout << "Line 6\\n";
    cout << "Line\\t7\\n";
    cout << "Line\\tt7\\n";
    return 0;
}

```

Sequences Characters\" && g++ app.cpp -o app && "f:\Programming\4- Elzero Web School\#2 - Fundamentals Of Programming With C ++#015 - Escape Sequences Characters\app Line 1 Line 2 Line 3\\ Line "4" Line '5' Line 6 Line 7 Line 7



٩- \b وهي ال white space أي يقوم بمسح الذي قبله إذا كان حرف او back space

```
30     cout << "Line\b8"; // Lin8
31     return 0;
32 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Lin8

١٠- \a هي Beep أو تظهر عند الطباعة في ال stream

Carriage Return \r - ١١

```
32     cout << "Fady\rAlamir"; // Alamir
33     return 0;
34 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Alamir

مثلاً تقوم بطباعة Fady\rAlamir ثم يرجع الى Carriage Return ثم يقوم بعمل override على Fady ويطبع ما بعد \r

Fady ← Alamir مكانتها

#016 - Data Types - What Is Data?

Data Types

- What Is Data?
- Data Examples In Real Life
 - Integer => 5000, 10, -100
 - String => "Elzero Web School", "Osama Elzero", "100A"
 - Boolean => true, false OR yes, no OR 1, 0 OR on, off
 - Float => 18.5, 1900.50
 - Array => ["Osama", "Ahmed", "Sayed", "Mahmoud"]
- Why We Choose Data?
- Size
- Operation

What Is Operand?

- The Part Of an Instruction Representing The Data Manipulated by The Operator



١- البيانات **Data**: مثل تطبيق برمجي لإدارة مدرسة وهو عبارة عن **Code**

البيانات: هو مجموعة التعليمات التي تُعطي للكمبيوتر لتنفيذ الفكرة البرمجية على

الـ Data: أسماء الطلاب والمدرسين وبياناتهم

٢- أنواع البيانات :Data Types

١. **Integer:** العدد الصحيح بدون كسور مثل salary المدرس ورقم المكتب

٢. String: النصوص مثل اسم المدرسة واسم المدرس واسم الفصل

٣- Boolean : تعبّر عن أي إجابة بها Yes, No أو أي On, Off switch .

Float (floating point number) OR Double .

الأرقام التي بهاكسور

٥. **Array:** وضع مجموعة من الأسماء او الارقام في قائمة

٣- لماذا نقوم بتحديد أنواع البيانات:

١. لأن كل نوع من أنواع البيانات له حجم في ذاكرة الكمبيوتر

```
21 int main()
22 {
23     int num = 10;
24     string name = "Fady";
25     bool status = true;
26
27     cout << sizeof(num) << "\n";
28     cout << sizeof(name) << "\n";
29     cout << sizeof(status) << "\n";
30 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Programming With C++\#016 - Data Types - What Is Data\app

4
32
1

Operation .٢: كل نوع من أنواع البيانات اختاره على أساس العمليات المناسبة له، لا ينفع أن تقوم بقسمة رقم على string

```
35     cout << num_one / name; // Error
36
37     return 0;
38 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Programming With C++ #016 - Data Types - What Is Data\app
app.cpp: In function 'int main()':
app.cpp:35:21: error: no match for 'operator/' (operand types are 'int' and 'std::string')
 cout << num_one / name; // Error
 ^~~~
 |
 int std::string [aka std::basic_string<char>]



٣: هو جزء من التعليمات البرمجة يمثل الـ Data Operand التي يتم التعامل معها.

#017 - Data Types, Sizes And Memory

١: ذاكرة الكمبيوتر بها مجموعة من الأماكن تخزن بها البيانات.

٢ - أحجام البيانات: Data Sizes

١ . Bit - [Bi]nary Digi[t]: أصغر وحدة تخزين يُخزن بها الـ 0, 1 فقط.

٢ . Byte: عبارة عن مجموعة مكونة من 8bits ويُخزن به حرف واحد.

Tera Byte => 1024 Gigabytes .٣

Giga Byte => 1024 Megabytes .٤

Mega Byte => 1024 Kilobytes .٥

Kilo Byte => 1024 Byte .٦

٣ - أنواع البيانات وأحجامها: Data Types With Size

١ . int => 2 Or 4 Bytes => Will Cover Later Way .١

٢ . float => 4 Bytes [18.5656565656]

٣ . double => 8 Bytes [18.5656565656]

٤: أي الجزء الذي بعد الكسر يُسمى مكون الكسر. أي الجزء الذي يلي الكسر يُسمى Number . Fractional Number - الفرق بينFloat & Double -

١ . Float : يمكن استخدام ٧ أرقام ككسور

٢ . Double : يمكن استخدام ١٥ رقم ككسور أي Double .

٤ . char => 1 Byte => "A" "X" "9"

٥ . boolean => 1 Byte => true, false



٤- القصة التي تحدث وراء انشاء المتغير "خلف الكواليس"

١- الإعلان عن متغير – Declare A Variable

٢- إخبار الكمبيوتر بأن يقوم بجز عدد معين من ال **Bytes** في ال **Memory** بناءً على نوع البيانات ثم يقوم بوضع القيمة التي سوف تقوم بوضعها في المكان الذي قمت بتحديده في ال **Memory**.

٣- يمنع إضافة أي نوع من البيانات الأخرى غير التي قمت بتحديدها.

٤- يمكن أن تجعل ال **compiler** يقوم باستنتاج نوع المتغير من القيمة التي أعطيتها للمتغير عن طريق وضع كلمة **auto** قبل أسم ال **variable**.

```
51 | auto num = 10;
52 | cout << num;
53 | return 0;
54 | }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
10

ولكن عندما لا تقوم بإعطاء قيمة لها سوف يحدث **error** لأنه لن يكون قادر عن تحديد نوع المتغير

```
51 | auto num;
52 | cout << num << "\n";
```

PROBLEMS 2 OUTPUT TERMINAL DEBUG CONSOLE
app.cpp: In function 'int main()':
app.cpp:51:3: error: declaration of 'auto num' has no initializer
| 51 | auto num;
| | ^~~~

وحتى إن قمت بتعديل القيمة لن يفهمها ال **compiler**

```
51 | auto num;
52 | num = 10;
53 | cout << num;
54 | return 0;
55 | }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
types, SIZES AND MEMORY\app\app.cpp app dd 1..01 Programming
Of Programming With C++#017 - Data Types, Sizes And Memory\app
app.cpp: In function 'int main()':
app.cpp:51:3: error: declaration of 'auto num' has no initializer
| 51 | auto num;
| | ^~~~



٦- يمكن معرفة الـ Memory Location عن طريق وضع & - Ampersand قبل اسم المتغير بعد cout يجعل الـ compiler يقوم بطباعة موقع المتغير في الذاكرة.

```
53 |     int nums = 100;
54 |     cout << &nums;
55 |     return 0;
56 | }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
0x78d47ffd68

#018 - Data Types – Integer

– أنواع البيانات الأولية Primitive Data Types - ١

(100, -500, 0) : العدد الصحيح مثل الـ Integer .)

٢. عند وضع true أو false لـ Int يعطي الـ Int القيمة ١ لـ true وصفر لـ false

```
23 |     int num_four = true;
24 |     int num_five = false;
25 |
26 |     cout << num_four << "\n";
27 |     cout << num_five;
28 |     return 0;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1
0

٣. لا يمكن طباعة حروف او characters او أي نوع اخر مع int بدلأً من الاعداد الصحيحة.

```
25 |     int num_six = "Fady";
26 |
27 | PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
28 | app.cpp: In function 'int main()':
29 | app.cpp:25:17: error: invalid conversion from 'const char*' to 'int' [-fpermissive]
30 |     25 |     int num_six = "Fady";
31 |           |           ^
32 |           |           |
33 |           |           const char*
```

٤. محدد في اللغة أكبر عدد موجب يمكن وضعه في المتغير int وأصغر عدد سالب

```
33 |     cout << INT_MIN << endl;
34 |     cout << INT_MAX << endl;
35 |     return 0;
36 | }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
-2147483648
2147483647



٥. يوجد بداخله الـ INT_MAX والـ INT_MIN يسمى header file limits.h

٦. أحجام أنواع البيانات

```
34 cout << INT_MAX << endl;
35
36 cout << sizeof(int) << " Byte" << endl;
37 cout << sizeof(char) << " Byte" << endl;
38 cout << sizeof(float) << " Byte" << endl;
39 cout << sizeof(double) << " Byte" << endl;
40 cout << sizeof(bool) << " Byte" << endl;
41
42 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
-2147483648
2147483647
4 Byte
1 Byte
4 Byte
8 Byte
1 Byte

٧. عند وضع عدد صحيح لمتغير نوعه int يقوم بتجاهل fractional number



```
42 int last_num = 10.5;
43 cout << last_num << endl;
44 return 0;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1 Byte
10

٨. الـ = تسمى Assignment Operator أي تقوم بـ الـ value للمتغير num_one أي تعين القيمة للمتغير.

```
18 int main()
19 {
20     int num_one = 100;
21     int num_two = -500;
22     int num_three = 0;
23     int num_four = true.
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
100

#019 - Data Types - Float And Double

١- نقوم باختيار نوع البيانات ولا نقوم بوضعها في أي نوع من البيانات بسبب الـ .Memory المحددة في الـ Bytes وحجم الـ Performance



٢- من الممكن وضع عدد صحيح في double للاحتساب إذا قمنا بالتعديل عليه وجعله كسور

```
20 |     double dob = 10;
21 |     dob = 20.5;
22 |     cout << sizeof(dob) << endl; // 8 Bytes
23 |     cout << dob << endl;           // 20.5
24 |
25 | }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
20
8
20.5
```

٣- لنقوم بجعل ال compiler يتعامل مع الأرقام على أنها float نقوم بوضع f بعد الرقم

```
cout << dob << endl;          // 20
                                (double)(9.5)
float fl = 10.5f + 9.5;
```

→

```
cout << dob << endl;
                                (float)(10.5F)
float fl = 10.5f + 9.5f;
```

ونقوم بجعل ال compiler على التعامل مع هذه الأرقام على أنها float بدلاً من ال double لأنه لو تعامل معها على أنها double ستكون أبطأ من لو تعامل معها على أنها

float

٤- وعند وضع ال f بعد الرقم مع نوع المتغير auto يجعلها أيضاً float

```
27 |     cout << fl << endl;
28 |     (float)(10.5F)
29 |     auto mix = 10.5f;
30 |     cout << mix << endl;
31 |
32 |     return 0;
33 | }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
10.5
```

#020 - Data Types - Char And ASCII

١- نقوم بتخزين حرف واحد فيه فقط ويحجز single quotes من الرام والقيمة توضع داخل 1 Byte

٢- نقوم بوضع (%) وينتج لنا ال ASCII Number لعلامة %: Type Casting

```
31 |     cout << int('%') << "\n"; // 37
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
37
```



وعندما نقوم بوضع `double quotes` لأن `error` يحدث لأن `cout` يأخذ `array of characters` لا ينفع.

٣- يمكن إنتاج الحرف باللغة الإنجليزية عن طريق رقمه في `ASCII Table` هكذا

```
37 cout << char(81) << "\n"; // Q
38
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Q

#021 - Data Types - Bool And Void

:Boolean - ١

١. قيم ال `Boolean` عبارة عن `true, false` جميع الحروف `small ture, false` لأن الحروف `أو أي إجابة بها Yes, No` أو أي `On, Off` به `sensitive`.

٢. نستخدم `true, false` ولا نقوم باستخدام `0, 1` لأن ال `Boolean` لد `نستخدم true, false فقط لكن ال 0, 1 عبارة عن integer تقوم بحجز .1, 0` لهذا سوف يكون ال `Performance` أقل إذا استخدمنا ال `0, 1`.

```
32 cout << test_one << endl; // 1
33 cout << test_two << endl; // 0
34 int num = 1;
35 cout << sizeof(test_one) << endl; // 1 Byte
36 cout << sizeof(num) << endl; // 4 Bytes
37 return 0;
38 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1
0
1
4

٣. لا نستخدم في ال `Boolean` فقط `true, false` ولكن من الممكن أن نقوم باستخدام معادلات أو معاذلة أو عملية معينة وينتج منها قيمة.

```
30 bool test_one = 10 > 5; // Yes => True => 1
31 bool test_two = 10 > 100; // No => False => 0
32 cout << test_one << endl; // 1
33 cout << test_two << endl; // 0
34 return 0;
35
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1
0



٤. عند وضع قيمة لـ **Boolean** يكون الناتج **True** أي ١ وعندما لا نضع قيمة يكون الناتج أي . **False**

```

37  bool num_one = 100;
38  bool num_two = -100;
39  bool num_three = 0;
40  cout << num_one << endl;      // 1
41  cout << num_two << endl;      // 1
42  cout << num_three << endl;    // 0
43  return 0;
44 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

1
1
0
```

٢ - **Void** : أي قيمته قيمة فارغة

وهي عبارة عن **Function** تقوم بدور معين لكن لا تقوم بإرجاع قيمة بحيث تقوم بفعل معين في التطبيق تربط شيء بشيء لكن لا تقوم بإرجاع قيمة لك، ولا تحتاج أن تقوم بعمل **return 0;** في نهاية ال **.function**

```

void without_value()
{
    // Nothing To Return
}
```

#022 - Data Types - Modifiers And Type Alias

١- نقوم باستخدام **short int** عندما لا نحتاج لـ **max int** مثل العمر حيث أن العمر لن يصل إلى هذا الرقم **#define SHRT_MAX 32767** وتقليل حجم ال Bytes المحفوظة في الرايم من ٤ Bytes لـ ٢ مما يزيد من **performance** البرنامج.

ال **short int** هو أقصر من ال **integer** الطبيعي.

```

34  short int new_age = 300;
35  cout << sizeof(new_age) << "\n"; // 2 Bytes
36
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

2
```

ويمكن كتابة **short** فقط بدون **int** هكذا

```

37  short last_age = 300;
38  cout << sizeof(last_age) << "\n"; // 2 Bytes
39
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

2
```

٢- **long int** : والتي ممكن ان تكتب **long** فقط، في نظام ال **windows** وال **vs compiler** تقوم بحجز 4 Bytes في الرايم مثل ال **int** الأساسي ولكن في بعض الأجهزة وال **compilers** الأخرى يقوم بحجز .8 Bytes



3 - **long long**: يمكنك من استخدام رقم بهذا الحجم
لذا يمكننا التحكم في الـ **modifier**

4 - **integer**: مثل الـ **integer modifier** يمكننا التحكم به إما أن يكون رقم صغير أو يكون رقم كبير.

5 - **Signed [int, char]**: عن طريقه نخبر الـ **compiler** بأن نخزن أرقام موجبة وسالبة
وصفر وهذا الوضع الطبيعي الـ **Default** للـ **integer, character**
لأن لها أرقام "النظام الذي نعبر به عن الـ **characters**" أي لها **ASCII Value** بأرقام "بأرقام

6 - **Unsigned [int, char]**: يخزن بها أرقام موجبة فقط

```
59  unsigned int num_five = -10;
60  cout << num_five << "\n";
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
4294967286
```

7 - **Type Alias**: أسم مستعار للـ **Type** الموجود مسبقاً باستخدام **using, typedef**

```
64  typedef long long int bignum;
65
66  bignum my_number = 100010001000;
67  cout << my_number << "\n";
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
100010001000
```

```
62  using bignum = long long int;
63
64  bignum my_number = 100010001000;
65  cout << my_number << "\n";
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
100010001000
```

#023 - Data Types - Modifiers And Type Alias

Data Types

- Type Conversion
 - Convert Data of One Type To Another
- Implicit Conversion
 - Conversion Is Done Automatically By The Compiler
- Explicit Conversion AKA Type Casting
 - Conversion Is Done By The Programmer
- Data Loss
 - When Larger Type Is Converted To Smaller Type



١ - تحويل أنواع البيانات: Type Conversion

أي تحويل البيانات من نوع آخر، ويوجد نوعين من التحويل

١ . التحويل الضمني: Implicit Conversion

التحويل يتم تلقائياً عن طريق الـ compiler

Ex:

```
22 int a;
23 double b = 20.5;
24 a = b; // Compiler Will Convert Double Value Then Assign
25 cout << a << "\n"; // 20
26 cout << sizeof(a) << "\n"; // 4 Bytes
27 cout << " " << "\n";
28 cout << " " << "\n";
29 cout << " " << "\n";
30 cout << " " << "\n";
31 cout << " " << "\n";
32 cout << " " << "\n";
33 cout << " " << "\n";
34 cout << " " << "\n";
35 cout << " " << "\n";
36 cout << " " << "\n";
37 cout << " " << "\n";
38 cout << " " << "\n";
39 cout << " " << "\n";
40 cout << " " << "\n";
41 cout << " " << "\n";
42 cout << " " << "\n";
43 cout << " " << "\n";
44 cout << " " << "\n";
45 cout << " " << "\n";
46 cout << " " << "\n";
47 cout << " " << "\n";
48 cout << " " << "\n";
49 cout << " " << "\n";
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Code

20
4

ملحوظة: عندما نقوم بجمع رقمين صحيح وكسر سوف يقوم الـ compiler بفرض الـ **Explicit Conversion** على العملية لذا سوف نقوم باستخدام الـ **double**

```
37 int e = 20;
38 double f = 20.5;
39 cout << e + f << "\n"; // 20.5 + 20 = 40.5
40 cout << sizeof(e + f) << "\n"; // 8 Bytes
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

40.5
8

٢ - التحويل الصريح: Explicit Conversion

التحويل يتم عن طريق الـ **programmer** وتسمي هذه العملية احياناً **Type Casting**

Ex:

```
44 int g = 20;
45 double h = 20.5;
46 cout << g + (int)h << "\n"; // 20 + 20 = 40
47 cout << g + int(h) << "\n"; // 20 + 20 = 40
48 cout << sizeof(g + (int)h) << "\n"; // 4 Bytes
49 cout << sizeof(g + int(h)) << "\n"; // 4 Bytes
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

40
40
4
4

والطريقة الثانية للـ **Type Casting** هنا في المثال السابق تسمى **function like**



ملحوظة: عند تحويل نوع من أنواع البيانات يحدث فقدان للبيانات، وهذا منطقي وطبيعي لأن عندما نقوم بتحويل مثلاً من `short int` إلى `long long` من الطبيعي أن يحدث لهذه البيانات فقدان أي الرقم هيصغر.

#024 - Operators - Arithmetic Operators

Operators & Operands

"Symbols To Operate On Data"

- Arithmetic Operators

"For Mathematical Operators"

--- + => Addition

--- - => Subtraction

--- * => Multiplication

--- / => Division

--- % => Modulo => Reminder After Division

What Is Operand?

- The Part Of an Instruction Representing The Data Manipulated by The Operator

Operators - ١: هي عبارة عن الرموز التي تقوم بعمل عمليات معينة على البيانات.

Operands - ٢: هي البيانات الخاصة بنا الموجودة التي يتعامل معها كل operator منه يكمل بعض.

Arithmetic Operators - ٣: operators الخاصة بالعمليات الحسابية.

```
23 cout << 10 + 10 << "\n";
24 cout << 10.5 + 9.5 << "\n";
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
20
20
```

:Addition . ١

```
33 cout << 100 - 50 << "\n";
34 cout << 100 - -50 << "\n";
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
50
150
```

:Subtraction . ٢



:Multiplication .٣

36	<code>cout << 10 * 5 << "\n";</code>
PROBLEMS	OUTPUT

50	

:Division .٤

38	<code>cout << 20 / 5 << "\n";</code>
39	<code>cout << 12 / 5 << "\n";</code>
40	<code>cout << 12.5 / 5.5 << "\n";</code>
PROBLEMS	OUTPUT

4	
2	

38	<code>cout << 20 / 5 << "\n";</code>
39	<code>cout << 12.5 / 5.5 << "\n";</code>
PROBLEMS	OUTPUT

4	
2.4	

ملحوظة: يجب وضع `f`. لترمز عن الـ `float` لكي ينتج القيمة بالكسر لأن ناتج قسمة أي `int/int` ينتج `int`.

وأيضاً لو فيه رقم واحد فيهم `float` يكون الناتج بالكسر أي `float`

38	<code>cout << 20 / 5 << "\n";</code>
39	<code>cout << 12 / 5 << "\n";</code>
40	<code>cout << 12.5 / 5.5 << "\n";</code>
41	<code>cout << 12.5 / 5 << "\n";</code>
PROBLEMS	OUTPUT

4	
2	
2.4	
2.4	

٥. Modulus/Modulo: هو باقي القسمة أي الرقم الذي نأخذه من الرقم اللي معانا حتى يكون عدد صحيح يُقسم على الرقم الآخر.

47	<code>cout << 20 % 5 << "\n";</code>
48	<code>cout << 21 % 5 << "\n"; // 1</code>
49	<code>cout << 24 % 5 << "\n"; // 1</code>
PROBLEMS	OUTPUT

0	
1	
4	

ملحوظة: الـ `Modulus` لا يتعامل إلا مع `integer`

50	<code>cout << 24.5 % 5 << "\n";</code>
PROBLEMS	OUTPUT

app.cpp: In function 'int main()':	
app.cpp:50:16: error: invalid operands of types 'double' and 'int' to binary 'operator%'	
50	<code> cout << 24.5 % 5 << "\n";</code>
	~~~~ ^ ~
	double int



## #025 - Operators - Assignment Operators

### Operators & Operands

"Symbols To Operate On Data"

- Assignment Operators

"For Assigning Operators"

--- = Assign

--- += Addition

--- -= Subtraction

--- *= Multiplication

--- /= Division

--- %= Modulo => Remainder After Division

Assignment Operators - هي التي تقوم بعمل assign لل value على اليسار لل variable على اليمين.

1 - Assign : هي علامة ال= التي نستخدمها لوضع قيمة في variable هكذا

2 - Addition : معناها ان ال compiler يقوم باستخدام ال variable ويجمع

القيمة التي بعد ال+= على قيمة ال variable هكذا

```
29  a += 15;           // a = 10 + 15 = 25
30  cout << a << "\n"; // 25
31
```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL  
25

Subtraction - 2 : معناها ان ال compiler يقوم باستخدام ال variable ويطرح

القيمة التي بعد ال-= من قيمة ال variable هكذا

```
32  int b = 20;
33  // b = b - 10;        // b = 20 - 10 = 10
34  b -= 10;             // b = b - 10
35  cout << b << "\n"; // 10
```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL  
10



variable : معناها ان الـ compiler يقوم باستخدام الـ Multiplication -٤  
ويضرب القيمة التي بعد الـ  $=*$  في قيمة الـ variable هكذا

```
37 int c = 5;
38 // c = c * 10;           // c = 5 * 10 = 50
39 c *= 10;                // c = c * 10 = 50
40 cout << c << "\n";
```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

50

Division -٥ : معناها ان الـ compiler يقوم باستخدام الـ variable ويقسمه على القيمة التي بعد الـ  $/$  هكذا

```
42 int d = 30;
43 // d = d / 5;           // d = 30 / 5 = 6
44 d /= 5;                // d = d / 5 = 6
45 cout << d << "\n";
```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

25

Modulus -٦ : معناها ان الـ compiler يقوم باستخدام الـ variable ويقسمه على القيمة التي بعد الـ  $\%$  وينتج الباقي في الـ stream هكذا

```
47 int e = 40;
48 // e = e % 9;           // e = 40 % 9
49 e %= 9;                // e = e % 9
50 cout << e << "\n";
```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

0

## #026 - Operators - Increment And Decrement Operators

Increment and Decrement Operators - العوامل المسؤولة عن الزيادة والنقصان

Pre - ١: شيء يحدث قبل عملية الـ execute

```
21 int views = 0;
22 cout << ++views << "\n"; // 1
23 cout << views << "\n"; // 1
```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

1  
1

هذا يسمى Pre Increment بحيث يزود الـ 1 وبعدها يقوم بطباعة الـ 1



## Post - ٢: شيء يحدث بعد عملية ال execute

```
21 int likes = 0;
22 cout << likes++ << "\n"; // 0
23 cout << likes << "\n"; // 1
```

PROBLEMS    **OUTPUT**    DEBUG CONSOLE    TERMINAL

```
0
1
```

هذا يسمى Post Increment بحيث يطبع ال 0 وبعدها يقوم بزيادة 1 عليها

-٣: يقوم بطباعة ال 0 وبعدها يقوم بطرح 1

-٤: يقوم بطرح 1 وبعدها يقوم بطباعة ال -1

```
25 int likess = 0;
26 cout << likess-- << "\n"; // 0
27 cout << likess << "\n"; // -1
28
29 int viewss = 0;
30 cout << --viewss << "\n"; // -1
31 cout << viewss << "\n"; // -1
```

PROBLEMS    **OUTPUT**    DEBUG CONSOLE    TERMINAL

```
0
-1
-1
-1
```

## #027 – Operators – Comparison Operators

- Comparison Operators

"For Comparing Values"

- - - == Equal
- - - != Not Equal
- - - > Greater Than
- - - < Less Than
- - - >= Greater Than Or Equal
- - - <= Less Than Or Equal

بعض واحياناً تسمى ال Operators المسؤولة عن المقارنة لمقارنة القيم - **Comparison Operators** -  
أي ال Operators Relational Operators التي يجعلك ترى  
علاقة القيم ببعضها البعض.



## تسخدم لمقارنة قيمتين بعض .Equal - ١

```
22 cout << (10 == 10) << "\n"; // 1 => True  
23 cout << (1000 == 100) << "\n"; // 0 => False
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1  
0

لمقارنة القيمتين وسؤال compiler هل هذه القيمة لا تساوي القيمة الأخرى .Not Equal - ٢

```
25 cout << (10 != 10) << "\n"; // 0 => False  
26 cout << (1000 != 100) << "\n"; // 1 => True
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

0  
1

للسؤال هل قيمة أكبر من قيمة أخرى .Greater Than - ٣

```
28 cout << (40 > 18) << "\n"; // 1 => True  
29 cout << (18 > 18) << "\n"; // 0 => False
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1  
0

للسؤال هل قيمة أصغر من قيمة أخرى .Less Than - ٤

```
31 cout << (16 < 18) << "\n"; // 1 => True
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1

للسؤال هل قيمة أكبر من او يساوي قيمة أخرى وعند تحقق شرط من الشرطين ينتج True .Greater Than Or Equal - ٥

```
33 cout << (18 >= 18) << "\n"; // 1 => True  
34 cout << (40 >= 18) << "\n"; // 1 => True
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1  
1

للسؤال هل قيمة أصغر من او يساوي قيمة أخرى وعند تحقق شرط من الشرطين ينتج True .Less Than Or Equal - ٦

```
36 cout << (18 <= 18) << "\n"; // 1 => True  
37 cout << (40 <= 18) << "\n"; // 0 => False
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1  
0



## #028 - Operators - Logical Operators

- Logical Operators  
"For Logic Between Values"

--- && And  
--- || Or  
--- ! Not

هي ال Operators التي من خلالها نستطيع أن نفحص المنطق بين القيم الخاصة بنا "نطلب أكثر من طلب".

١ . ( && )And : يتكون من علامتين Ampersand أو تسمى "و" العطف ومعناها and نستخدمها لطلب طلبيين أو أكثر من طلب في سطر واحد ويجب تحقق جميع الطلبات.

```
cout << (age >= 18);
cout << (points >= 500);
```

→

23	cout << (age >= 18 && points >= 500) << endl; // 1 => True
PROBLEMS	OUTPUT
1	TERMINAL

25	int age = 16; int points = 800; cout << (age >= 18 && points >= 500) << endl; // 0 => False
PROBLEMS	OUTPUT
0	DEBUG CONSOLE

٢ . ( || )Or : يتكون من علامتين Pipe معناهم "أو" وتستخدم لطلب طلب أو أكثر من طلب في سطر واحد وإذا تحقق أي طلب من المطالب ينتج True

26	int age = 16; int points = 800; cout << (age >= 18    points >= 500) << endl; // 1 => True
PROBLEMS	OUTPUT
1	DEBUG CONSOLE

30	int age = 16; int points = 450; cout << (age >= 18    points >= 500) << endl; // 0 => False
PROBLEMS	OUTPUT
0	DEBUG CONSOLE

32	cout << (age >= 18    points >= 500) << endl; // 0 => False 33 cout << (100 == 10    50 == 10    20 == 10    10 == 10) << endl; // 1 => True
PROBLEMS	OUTPUT
0	DEBUG CONSOLE
1	TERMINAL



## ٣ .٢ (!): تقوم بعكس الحاجة أي "نفي النفي" ← أثبات

```
35 cout << (10 == 10) << endl; // 1 => True
36 cout << !(10 == 10) << endl; // 0 => Not True => False
37 cout << !(100 == 10) << endl; // 1 => Not False => True
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
1
0
1
```

### #029 - Operators – Precedence

- Operators Precedence

"Which One Has Higher Precedence"

Reference

--- Operators Precedence Table

Training

- Try Operators Yourself Before Browsing References

١ - **Operator Precedence**: أي أن Operator له الأولية أن يعمل قبل آخر إذا تواجدوا جميعهم في نفس السطر أو نفس العملية، وبعض الـ Operators لهم نفس الأولوية في العمل عند إذ يقوم الـ compiler بترجمة الكود من الشمال لليمين بالترتيب الطبيعي.

مثل هذا المثال: الضرب له الأولوية في العمل قبل Operator الجمع والطرح

```
24 cout << 10 + 5 * 5 << "\n";
25 // 5 * 5 = 25
26 // 10 + 25 = 35
27 cout << 10 - 5 * 5 << "\n";
28 // 5 * 5 = 25
29 // 10 - 25 = -15
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
35
-15
```

مثال ٢: الـ Operator الخاص بالضرب والـ Operator الخاص بالقسمة لهم نفس الأولوية لذا سوف تسير العملية من الشمال لليمين كالطبيعي

```
30 cout << 20 / 5 * 4 << "\n";
31 // 20 / 5 = 4
32 // 4 * 4 = 16
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
16
```



مثال ٣: الـ **Operator** الخاص بالضرب والـ **Operator** الخاص بالقسمة الاثنين لهم نفس الأولوية في العمل قبل الجمع لذا سوف تتم عملية الضرب والقسمة أولاً من الشمال لليمين ومن ثم عملية الجمع

```
33 cout << 10 + 20 / 5 * 4 << "\n";
34 // 20 / 5 = 4
35 // 4 * 4 = 16
36 // 10 + 16 = 26
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL  
26

**ملحوظة:** إذا أردت أن تتم العملية من الشمال لليمين بدون ترتيب الأوليات للـ **Operators** يجب أن تقوم بعزل عملية الجمع أو الطرح بوضعها في أقواس "parentheses"

```
37 cout << (10 + 5) * 5 << "\n"; // (15) * 5 = 75
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL  
75

## #030 - Control Flow - If Condition Introduction

: هو التحكم في تدفق البيانات الموجودة في التطبيق أو الكود البرمجي.

١. الـ **if** الخاص بها

```
if (Condition Is True)
{
    // Do Something
}
```

```
17 int age = 15;
18 cout << "Welcome\n";
19
20 if (age < 18) // True
21 {
22     cout << "Beware\n";
23 }
24
25 cout << "See You\n";
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL  
Welcome  
Beware  
See You

```
17 int age = 20;
18 cout << "Welcome\n";
19
20 if (age < 18) // False
21 {
22     cout << "Beware\n";
23 }
24
25 cout << "See You\n";
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL  
Welcome  
See You



## #031 - Control Flow – If, Else If, Else

1- If : هو الشرط الأول عند تتحققه ينفذ ولا ينظر لباقي الشروط else if, else

```
15 int main()
16 {
17     int age = 25;
18     int points = 450;
19     int rank = 4;
20
21     if (age >= 18)
22     {
23         cout << "Welcome Your Age Is Ok\n";
24     }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Welcome

if : يتحقق وينفذ عندما لم يتحقق الشرط الأول else if - ٢

```
17 int age = 15;
18 int points = 800;
19 int rank = 4;
20
21 if (age >= 18)
22 {
23     cout << "Welcome Your Age Is Ok\n";
24 }
25 else if (points > 500)
26 {
27     cout << "Welcome Your Points Is Ok\n";
28 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Welcome Your Points Is Ok

```
17 int age = 15;
18 int points = 450;
19 int rank = 8;
20
21 if (age >= 18)
22 {
23     cout << "Welcome Your Age Is Ok\n";
24 }
25 else if (points > 500)
26 {
27     cout << "Welcome Your Points Is Ok\n";
28 }
29 else if (rank > 5)
30 {
31     cout << "Welcome Your Rank Is Ok\n";
32 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Welcome Your Rank Is Ok

if, else if : يتحقق وينفذ عندما لما يتحقق كلاً من الشرطين else - ٣

```
17 int age = 15;
18 int points = 450;
19 int rank = 4;
20
21 if (age >= 18)
22 {
23     cout << "Welcome Your Age Is Ok\n";
24 }
25 else if (points > 500)
26 {
27     cout << "Welcome Your Points Is Ok\n";
28 }
29 else if (rank > 5)
30 {
31     cout << "Welcome Your Rank Is Ok\n";
32 }
33 else
34 {
35     cout << "Iam Sorry\n";
36 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Iam Sorry



## #032 - Control Flow - Nested If Conditions

**أي حالات شرط متداخلة أي Nested If Conditions**

أخرى، تستخدم لعمل filtration لمجموعة من البيانات أي ذا condition هذه البيانات حيث أول شرط نقوم بعمل filter لبيانات معينة ومن ثم نبدأ التعامل مع البيانات المتبقية بسهولة عن طريق الشروط الداخلة Nested Conditions الموجودة داخل الشرط الأساسي الذي قمنا من خلاله بعمل filter للبيانات.

مثل هذا المثال: أول شرط قمنا بعمل filter للأشخاص الذي ذو سن أصغر من ١٨ سنة

ومن ثم يتبقى مجموعة من البيانات نتعامل معها بواسطة Nested Conditions

```

16  if (age >= 18)
17  {
18      cout << "Welcome Your Age Is Ok\n";
19      if (points >= 1000)
20      {
21          cout << "You are VIP\n";
22      }
23 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Welcome Your Age Is Ok  
You are VIP

## #033 - Control Flow - Ternary Conditional Operator

**هو عبارة عن الـ If Condition المختصرة Ternary Operator**

Syntax

(Condition Is True) ? True : False;

```

14  int age = 25;
15
16  if (age >= 18)
17  {
18      cout << "Your Age Is OK\n";
19  }
20  else
21  {
22      cout << "Your Age Is Not OK\n";
23  }
24
25  cout << (age >= 18 ? "Your Age Is OK\n" : "Your Age Is Not OK\n");
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Your Age Is OK  
Your Age Is OK

ومن الممكن ان نقوم بعمل assign للقيمة الناتجة من هذا الشرط لمتغير معين

```

27  string msg = age >= 18 ? "Your Age Is OK\n" : "Your Age Is Not OK\n";
28
29  cout << msg;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Your Age Is OK



## #034 - Control Flow - Nested Ternary Operator

### Control Flow

- Nested Ternary Operator
- Alternate Syntax For If Condition

### Syntax

```
(Condition Is True) ? True : False;
```

**Nested Ternary Operator - ١**  
في الـ **Code** يطبع الناتج المكتوب

وفي الـ **else** في حالة الـ **False** تكون الـ **True** عبارة عن **Nested Condition** شرط آخر وليس طباعة كلمة فقط

```
34 cout << (age >= 18 ? "OK\n" : (points >= 500 ? "OK Because Of Points\n" : "No Age Or Points\n"));
35 // cout << (points >= 500 ? "OK Because Of Points\n" : "No Age Or Points\n");
36
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Code

OK Because Of Points

```
34 cout << (age >= 18 ? "OK\n" : (points >= 500 ? "OK Because Of Points\n" : "No Age Or Points\n"));
35 // cout << (points >= 500 ? "OK Because Of Points\n" : "No Age Or Points\n");
36
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Code

No Age Or Points

ويمكن إضافة الـ **True** أو الـ **False** في الـ **Nested Ternary Operator**

```
34 cout << (age >= 18 ? "OK\n" : (points >= 500 ? "OK Because Of Points\n" : "No Age Or Points\n"));
35 cout << (age >= 18 ? (points >= 500 ? "OK Because Of Points\n" : "No Age Or Points\n") : "No Age\n");
36
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Code

No Age Or Points  
No Age

**٢ - عندما يكون الـ **one line** عبارة عن **Conditions** في الـ **Block of Code** هكذا فلا تحتاج إلى كتابة الـ **Curly Brackets****

```
40 if (age >= 18)
41 | cout << "OK\n";
42 else
43 | cout << "Not OK\n";
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Not OK



ولكن إذا تكون من سطرين هكذا بدون الـ curly brackets يحدث error

```

40  if (age >= 18)
41    cout << "OK\n";
42    cout << "OK\n";
43 else
44   cout << "Not OK\n";
45
46 return 0;

```

## #035 - Condition Trainings - Create Four Application

١- من الممكن في الـ cin المدخلات التي تطلب من الـ user والمخرجات cout ان نقوم

إضافة ٣ متغيرات بجانب بعض بهذه الطريقة cout << a << b << c; cin >> a >> b >> c;

### Done Applications

## #036 - Control Flow - Switch Case

معناها التحويل أو التبديل بين شيء وشيء آخر

الـ Syntax -

```

switch (expression)
{
case /* constant-expression */:
/* code */
break;

default:
break;
}

```

- الفرق بين switch ، if -

```

if (day == 1)
{
  cout << "Open From 08:00 To 14:00";
}
else if (day == 2)
{
  cout << "Open From 08:00 To 14:00";
}
else if (day == 3)
{
  cout << "Open From 10:00 To 16:00";
}
else
{
  cout << "Closed";
}

```



```

switch (day)
{
case 1:
  cout << "Open From 08:00 To 14:00";
  break;
case 2:
  cout << "Open From 08:00 To 14:00";
  break;
case 3:
  cout << "Open From 10:00 To 16:00";
  break;
default:
  cout << "Closed";
}

```



## - طريقة عملها أو الـ **Cycle** الخاصة بالـ **day**

يمشي بالترتيب لو الشخص كتب 1 يبدأ ينفذ الـ **block of code** اللي بعده بعد كدا الـ **break** بتطلع برا الـ **block of code** تعمل **terminate** أي انهاء للـ **switch** وتنهي الـ **Cycle** في هذا الجزء وإذا لم يكن الـ **case** هو 1 يبدأ يدخل على الـ **case 2** وإذا لم يكن 2 أو 3 يرجع للـ **default**

- كملة **break;** اختيارية من الممكن أن لا نقوم بكتابتها ولكن في بعض الأمثلة لكن لا تنفع في المثال السابق

```
32 switch (day)
33 {
34     case 1:
35         cout << "Open From 08:00 To 14:00";
36         // break;
37     case 2:
38         cout << "Open From 08:00 To 14:00";
39         break;
40     case 3:
41         cout << "Open From 10:00 To 16:00";
42         break;
43     default:
44         cout << "Closed";
45 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE  
PS C:\Users\ss> & 'c:\Users\ss\vscode\extensions\ms-vscode.cpptools-1.15.4-win32-x64\debugAdapters\b1n4d1d-2\bpdb2.1if' --debgx=C:\Vsyst64\mingw64\bin\gdb.exe' '--interpreter=mi'  
Choose A Day From 1 To 25  
Open From 08:00 To 14:00Open From 08:00 To 14:00  
PS C:\Users\ss> []

- من الممكن دمج الحالتين مع بعضهم البعض

```
case 1:
case 2:
    cout << "Open From 08:00 To 14:00";
```

- الـ **switch** لا يقبل نوع بيانات غير **integer, character**

```
bu switch quantity not an integer gcc
float day
View Problem (Alt+F8) No quick fixes available
(day)
```

```
switch conversion from 'double' to 'int' in a converted constant expression gcc
{
    case 10.5:
}
```



## #037 - Switch Training - Create Three Application

ملحوظة: عندما لا نقوم بوضع الـ default في switch يقوم بإستدعاء المتغير الثابت

```
40 // App 2 => Discount Application
41
42 int price = 100;
43 int discount = 10;
44 int years;
45 cout << "Type The Number Of Years in Company\n";
46 cin >> years;
47
48 switch (years)
49 {
50     case 1:
51         discount = 20;
52         break;
53     case 2:
54         discount = 40;
55         break;
56     case 3:
57         discount = 80;
58         break;
59 }
60
61 cout << "The Price Is: " << price - discount << "\n";
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE  
Type The Number Of Years in Company  
5  
The Price Is: 99

## #038 - Array - What Is Array

### Arrays

- What is Array?

--- Collection Of Elements Of The Same Type

--- Placed in Contiguous Memory Locations

--- Referenced By Index Started From 0

- Why We Need Array?

- Creating Array Syntax

- Check Array Size

- Create Array Without Size

- المصفوفة :Array

1 - عبارة عن Collection مجموعة من البيانات من نفس النوع



## ٢- هذه العناصر توضع في أماكن متجاورة في الـ Memory

٣- كل عنصر من العناصر نصل إليه عن طريق الـ **Index** الخاص به والـ **Array** هي أي أن العنصر الأول في الـ **Array** الـ **index** الخاص به رقمه صفر

- استخدام الـ **Array** -

تستخدم لوضع العديد من البيانات كالأسماء والأرقام في مصفوفة واحدة وذ Access على جميعهم مع بعض بدلًا من وضع كل اسم أو رقم في متغير معين.

### Syntax of Array -

```
int nums[4] = {100, 200, 300, 400};
```

- حجم المصفوفة :**Array Size** -

يكون على حسب نوع البيانات ويكون حاصل ضرب عدد البيانات * حجم نوع البيانات بال Bytes

```
20 int nums[4] = {100, 200, 300, 400};
21 cout << sizeof(int) << "\n"; // 4 Bytes
22 cout << sizeof(nums) << "\n"; // 16 Bytes
23
24 double dos[4] = {100, 200, 300, 400};
25 cout << sizeof(double) << "\n"; // 8 Bytes
26 cout << sizeof(dos) << "\n"; // 32 Bytes
```

PROBLEMS 2 OUTPUT TERMINAL DEBUG CONSOLE

4  
16  
8  
32

ملحوظة: من الممكن عدم كتابة عدد عناصر الـ **Array** ويقوم الـ **Compiler** بإستنتاج عدد عناصر الـ **Array** تلقائيًا

```
int rands[3]
int rands[] = {100, 5000, 950};
```

ملحوظة ٢: ومن الممكن أيضًا كتابة الـ **Array Syntax** بدون علامة = بدون أي error

```
int rands[] {100, 5000, 950};
```



## #039 - Array - Access Elements & Memory Location

### Arrays

- Access Array Elements
- Check Element Location

١- لنقوم بعمل access على عنصر من عناصر الـ **Array** يكون عن طريق أسم الـ **Array** والـ **index** للعنصر

```
24 int nums[]{100, 200, 300};  
25 cout << "First Element: " << nums[0] << "\n";  
26 cout << "Last Element: " << nums[2] << "\n"; // Number Of Elements - 1  
  
PROBLEMS 2 OUTPUT TERMINAL DEBUG CONSOLE  
First Element: 100  
Last Element: 300
```

٢- يمكن الحصول على الـ **Memory Location** لعنصر من عناصر الـ **Array** عن طريق علامة الـ **&** and نضعها قبل أسم الـ **Array[index]**

```
24 cout << "Location: " << &nums[0] << "\n";  
25 cout << "Location: " << &nums[1] << "\n";  
26 cout << "Location: " << &nums[2] << "\n";  
  
PROBLEMS 2 OUTPUT TERMINAL DEBUG CONSOLE  
Location: 0x6ac3bffd54  
Location: 0x6ac3bffd58  
Location: 0x6ac3bffd5c
```

## #040 - Array - Add And Update Elements

### Arrays

- Declare Empty Array
- Add Elements To Array
- Update Array Elements
- Get Length Of Array With Sizeof



## Declare Empty Array & Add Elements To Array & Update Array Elements - ١

```

16 int nums[4];
17
18 nums[3] = 400; // Last Element
19 nums[2] = 300; // Third Element
20 nums[0] = 100; // First Element
21 nums[1] = 200; // Second Element
22
23 cout << "Elemnt 1: " << nums[0] << "\n";
24 cout << "Elemnt 2: " << nums[1] << "\n";
25 cout << "Elemnt 3: " << nums[2] << "\n";
26 cout << "Elemnt 4: " << nums[3] << "\n";

```

PROBLEMS 2 OUTPUT TERMINAL DEBUG CONSOLE

```

Elemnt 1: 100
Elemnt 2: 200
Elemnt 3: 300
Elemnt 4: 400

```

**ملاحظات:**

١ - لا يلزم تعديل قيم الـ **indexes** بالترتيب

٢ - عندما لا نقوم بإعطاء قيمة ويكون الـ **Index** فارغ ينتج في الـ **Terminal** رقم

**Random**

```

19 int nums[4];
20
21 nums[3] = 400; // Last Element
22 nums[2] = 300; // Third Element
23 // nums[0] = 100; // First Element
24 nums[1] = 200; // Second Element
25
26 cout << "Elemnt 1: " << nums[0] << "\n";

```

PROBLEMS 2 OUTPUT TERMINAL DEBUG CONSOLE

```

Element 1: 2099648152

```

٣ - عند تحديد الـ **Index** للمرة الثانية عند استخراجه ينتج في الـ **Terminal** الرقم

**الجديد بعد التحديث**

```

26 nums[1] = 1000; // Second Element
27
28 cout << "Element 2: " << nums[1] << "\n";

```

PROBLEMS 2 OUTPUT TERMINAL DEBUG CONSOLE

```

Element 2: 1000

```

## Get Length Of Array With sizeof - ٢

```

30 int anums[] = {100, 200, 300, 400, 500, 600}; // 24 / 4 = 6
31 cout << "Array Elements Count Is " << sizeof(anums) / sizeof(anums[0]) << "\n";

```

PROBLEMS 2 OUTPUT TERMINAL DEBUG CONSOLE

```

Array Elements Count Is 6

```



## #041 - Array - Two Dimensional Array

### Arrays

- Two Dimensional Arrays AKA [2D Array]

### Search For

- Matrix Operations
- 3D Array

أي جدول به **Table with rows and columns** : تعتبر **Two Dimensional Array - صفوف وأعمدة**.

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

```
14 int main()
15 {
16     int points_a[3] = {1, 2, 3};
17     int points_b[3] = {4, 5, 6};
18     int points_c[3] = {7, 8, 9};
19
20     // Good Practice
21     int points[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
22     cout << points[1][2] << "\n"; // 6
23     cout << points[2][0] << "\n"; // 7
24     cout << points[2][2] << "\n"; // 9
25
26     return 0;
27 }
```

PROBLEMS 2 OUTPUT TERMINAL DEBUG CONSOLE

6  
7  
9



```
// Bad Practice
int points[3][3] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
cout << points[1][2] << "\n"; // 6
cout << points[2][0] << "\n"; // 7
cout << points[2][2] << "\n"; // 9
```

**ملحوظة:** يجب أن يكون الـ `const int rows` ، `columns` ثابتين حتى لا يحدث `error` لأن إذا لم يكونوا ثابتين فيمكن تحدث صفات وأعمدة الـ `Array` ولا ينفع حدوث هذا

```
// Good Practice

const int rows = 3;
const int columns = 3;

int points[rows][columns] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
cout << points[1][2] << "\n"; // 6
cout << points[2][0] << "\n"; // 7
cout << points[2][2] << "\n"; // 9
```

## #042 - Array - Class Array

### 1 - طريقة أخرى لإنشاء الـ `Array`

**Syntax:** `Template<Type, Size> Identifier;`

```
19 int main()
20 {
21     // int points[4] = {1, 2, 3, 4}; // C-Style Array
22     array<int, 4> points = {1, 2, 3, 4};
23     cout << points[0] << "\n";
24     cout << points[1] << "\n";
25     cout << points[2] << "\n";
26     cout << points[3] << "\n";
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

1  
2  
3  
4

### 2 - لمعرفة حجم الـ `Array` نقوم باستخدام `.size()`

```
30 cout << "Elements Count: " << points.size() << "\n";
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Elements Count: 4



.fill -٣

تستخدم لجعل جميع الـ `indexes` الموجود في الـ `Array` بنفس القيمة

```
23 points.fill(10);
24 cout << points[0] << "\n";
25 cout << points[1] << "\n";
26 cout << points[2] << "\n";
27 cout << points[3] << "\n";
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
10
10
10
10
```

**ملحوظة:** عند وضع نوع آخر من البيانات في `fill`. مثل الـ `char` ينتج الـ `fill`.

```
25 points.fill('A');
26 cout << points[0] << "\n";
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
65
```

وعند وضع `Boolean value` ينتج للـ `True` رقم ١ ولـ `False` رقم ٢

```
28 points.fill(true);
29 cout << points[0] << "\n";
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
1
```

## #043 - Array - Methods Discussions

.front() -١

تقوم باستدعاء أول عنصر في الـ `Array`

```
24 array<int, 4> nums = {100, 200, 300, 400};
25 cout << nums[0] << "\n";
26 cout << nums.front() << "\n";
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
100
100
```

.back() -٢

تقوم باستدعاء آخر عنصر في الـ `Array`

```
24 array<int, 4> nums = {100, 200, 300, 400};
25 cout << nums[3] << "\n";
26 cout << nums.back() << "\n";
```

PROBLEMS 4 OUTPUT TERMINAL DEBUG CONSOLE

```
400
400
```



.at(element) -٣

تقوم باستدعاء element معين في المصفوفة من خلال رقم الـ index الخاص .به.

```
25 | array<int, 4> nums = {100, 200, 300, 400};  
26 | cout << nums.at(1) << "\n"; // 200  
PROBLEMS 5 OUTPUT TERMINAL DEBUG CONSOLE  
200
```

.size() -٤

تعطي عدد الـ elements الموجودة بالمصفوفة

```
25 | array<int, 4> nums = {100, 200, 300, 400};  
26 | cout << nums.size() << "\n"; // 4  
PROBLEMS 6 OUTPUT TERMINAL DEBUG CONSOLE  
4
```

.empty() -٥

للكشف عن المصفوفة إذا كانت فارغة أم لا وتعطي القيمة Boolean

```
26 | array<int, 4> nums = {100, 200, 300, 400};  
27 | cout << nums.empty() << "\n"; // 0 => False  
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE  
0
```

## #044 - Array Trainings - Guess The Number Game

Done

## #045 - String - What Is A String

١ - الـ String: هو أي شيء عبارة عن بيانات كنصوص مكتوبة مثل (الاسم – Full name – ، العنوان – Address ، المعرفة – About ، البريد الإلكتروني – Email ، الاسم المستعار – User name ،

ملاحظة: الـ String عبارة عن الـ Array Of Characters مجموعة من الـ Characters داخل الـ Array

```
int main()  
{  
    (const char [12])"Iam Dragon\n"  
    cout << "Iam Dragon\n";  
    return 0;  
}
```

وتكون مكونة من ١٢ حرفاً لان الـ string تنتهي بـ \0

```
cout << "Iam Dragon\n"; // 12 characters  
                                (const char [7])"Elzero"  
char name_a[] = "Elzero";  
cout << name_a << "\n"; // Elzero\0
```



## أي انهائها string ال Terminate بـ <= \0 تقوم

```
25 cout << "Iam\0 Dragon\n"; // 12 => Remember
26 cout << "\n";
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Iam

24 char name_a[] = "Elzero";
25 cout << name_a << "\n";           // Elzero\0
26 cout << sizeof(name_a) << "\n"; // 7
27 cout << name_a[0] << "\n";       // E
28 cout << name_a[5] << "\n";       // o
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Elzero
7
E
o

30 cout << int(name_a[6]) << "\n"; // \0 => Null => ASCII Value => 0
31 cout << int('`') << "\n";        // ` => Backspace => ASCII Value => 8
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
0
8
```

٢- الطريقة الأولى. بينها وبين array of characters الطريقة الـ **array** الطبيعية ولا يوجد بها اختلاف.

```
33 char name_b[] = {'E', 'l', 'z', 'e', 'r', 'o', '\0'};
34 cout << name_b << "\n";           // Elzero\0
35 cout << sizeof(name_b) << "\n"; // 7
36 cout << name_b[0] << "\n";       // E
37 cout << name_b[5] << "\n";       // o
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Elzero
7
E
o
```

٣- إنشاء الـ **string** عن طريق الـ **class** وهذه الطريقة لا تختلف عن البنية الأساسية وهو أيضاً يقوم بعمل الـ **array of characters** ولكن الفرق في الـ **class** يوجد **options**, **properties** تستخدم مع الـ **string** ذو أهمية.

**ملاحظة: الاختلاف الوحيد هو التخزين وسعته في ال RAM**

```
39 string name_c = "Elzero";
40 cout << name_c << "\n";           // Elzero\0
41 cout << sizeof(name_c) << "\n"; // 32
42 cout << name_c[0] << "\n";       // E
43 cout << name_c [5] << "\n";       // o
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Elzero
32
E
o
```



## ٤- وتختلف سعة التخزين في الـ RAM أيضاً بين الـ VS Code, Visual Studio

```
string name_c = "Elzero";
cout << name_c << "\n";           // Elzero\0
cout << sizeof(name_c) << "\n"; // 28
cout << name_c[0] << "\n";       // E
cout << name_c[5] << "\n";       // o
```

0
8
Elzero
7
E
o
Elzero
28
E
o

### #046 - String – Concatenating

## String

- Concatenating Strings
  - Normal Way
  - strcat => Include string.h
  - With +
  - append

في علوم الكمبيوتر هي نظرية ربط الـ strings ببعضهاConcatenating Strings - ١ البعض أو نظرية ربط الـ variables ببعضها.

ويمكن عمل الـ concatenate بالطريقة المعتادة

```
23  char fname[] = "Osama ";
24  char lname[] = "Elzero";
25
26  cout << fname << lname << "\n";
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Osama Elzero

أو بالـ methods هكذا "strcat" بإضافة "string.h" أو "cstring"

```
23  char fname[] = "Osama ";
24  char lname[] = "Elzero";
25
26  cout << strcat(fname, lname) << "\n";
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Osama Elzero



## أو عن طريق الـ +

```
23     string firstname = "Fady ";
24     string lastname = "Alamir";
25
26     cout << firstname + lastname << "\n";
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Fady Alamir

## أو عن طريق الـ `string.append` أي اللحق الـ `string` بالـ `string` الأخرى

```
23     string firstname = "Fady ";
24     string lastname = "Alamir";
25
26     cout << firstname.append(lastname) << "\n";
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Fady Alamir

## #047 - Loop With For

### Loop

- Loop With For
- Loop On Array

- الـ **Loop** أو التكرار: عن طريق نستطيع أن نقوم بتكرار **Block of code** أي عدد من المرات على حسب الفكرة

### :Loop With For Syntax -

```
31     for (int i = 0; i < 6; i++)
32     {
33         cout << i << "\n";
34     }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

0  
1  
2  
3  
4  
5

أول مرحلة في الـ `loop` الـ `0 = i` ثم هل الـ `6 > i` إذن يقوم بطبعها وبعدها يقوم بزيادة قيمتها 1 ثم يقوم بطباعية 1 وهكذا حتى يصل إلى الـ 5 ويقوم بطباعتها ويقوم بزيادة 1 ومن ثم الـ 6 ليست أصغر من الـ 6 إذن لا يقوم بطباعتها وتوقف هنا الـ `loop`



## شكل مبسط للـ loop لفهم الفكرة

```
18 int num = 0;
19 cout << num << "\n"; // 0
20 num++;
21 cout << num << "\n"; // 1
22 num++;
23 cout << num << "\n"; // 2
24 num++;
25 cout << num << "\n"; // 3
26 num++;
27 cout << num << "\n"; // 4
28 num++;
29 cout << num << "\n"; // 5
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
0
1
2
3
4
5
```

- تستخدم أيضاً طباعة عناصر ال Array



```
36 int nums[4] = {100, 200, 300, 400};
37 cout << nums[0] << "\n";
38 cout << nums[1] << "\n";
39 cout << nums[2] << "\n";
40 cout << nums[3] << "\n";
41
42 for (int index = 0; index < 4; index++)
43 {
44     cout << nums[index] << "\n";
45 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
100
200
300
400
100
200
300
400
```

## #048 - Loop With For - Advanced Syntax

يمكن طباعة عناصر ال Array هكذا

```
19 int nums[] = {100, 200, 300, 400, 500, 600};
20 int numsCount = sizeof(nums) / sizeof(nums[0]); // 6*4 = 24/4 = 6
21
22 for (int i = 0; i < numsCount; i++)
23 {
24     cout << nums[i] << "\n";
25 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
100
200
300
400
500
600
```



ملحوظة: لو ال Block of code في سطر واحد من الممكن إزالة ال curly brackets مثل ال if condition

```
for (int i = 0; i < numsCount; i++)
| cout << nums[i] << "\n";
```

ملحوظة ٢: من الممكن عدم كتابة ال initialize, condition, update داخل ال for وكتابتهم في مكان آخر.

### 1- Initialize

```
19 int nums[] = {100, 200, 300, 400, 500, 600};
20 int numsCount = sizeof(nums) / sizeof(nums[0]); // 6*4 = 24/4 = 6
21 int i = 0;
22
23 for (; i < numsCount; i++)
24 {
25     cout << nums[i] << "\n";
26 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

100  
200  
300  
400  
500  
600

### 2- Update

```
17 int nums[] = {100, 200, 300, 400, 500, 600};
18 int numsCount = sizeof(nums) / sizeof(nums[0]); // 6*4 = 24/4 = 6
19 int i = 0;
20
21 for (; i < numsCount;)
22 {
23     cout << nums[i] << "\n";
24     i++;
25 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

100  
200  
300  
400  
500  
600

### 3- Condition

نقوم باستخدام break; لعمل terminate loop

```
21 for (;;)
22 {
23     cout << nums[i] << "\n";
24     i++;
25     if(i == numsCount)
26     {
27         break;
28     }
29 }
```



## ملحوظة ٣: يجب مراعاة ترتيبهم الصحيح

```
17 int nums[] = {100, 200, 300, 400, 500, 600};  
18 int numsCount = sizeof(nums) / sizeof(nums[0]); // 6*4 = 24/4 = 6  
19 int i = 0;  
20  
21 for (;;)  
22 {  
23     i++;  
24     cout << nums[i] << "\n";  
25     if(i == numsCount)  
26     {  
27         break;  
28     }  
29 }  
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE  
200  
300  
400  
500  
600  
6
```

### #049 - Loop With For - Advanced Trainings

- طریقین لطباعة بعض الأرقام من الـ Array بواسطة الـ Loop زوجياً بالـ index

```
9 int main()  
10 {  
11     int nums[] = {100, 200, 300, 400, 500, 600};  
12     int numsSize = sizeof(nums) / sizeof(nums[0]);  
13  
14     // 100, 300, 500 - Method 1  
15     for (int i = 0; i < numsSize; i += 2)  
16     {  
17         cout << nums[i] << "\n";  
18     }  
19     cout << "\n";  
20  
21     // 100, 300, 500 - Method 2  
22     for (int i = 0; i < numsSize; i++)  
23     {  
24         cout << nums[i] << "\n";  
25         i++;  
26     }  
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE  
100  
300  
500  
  
100  
300  
500
```



## - طریقتین لطباعة العناصر تنازلياً

```
34 // 600, 500, 400, 300 - Method 1
35 for (int i = numsSize - 1; i > 1; i--)
36 {
37     cout << nums[i] << "\n";
38 }
39 cout << "\n";
40
41 // 600, 500, 400, 300 - Method 2
42 for (int i = 5; i > 1; i--)
43 {
44     cout << nums[i] << "\n";
45 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
600
500
400
300

600
500
400
300
```

## ٣- لوضع الـ update condition والـ initialize خارج الـ for

```
48 // Challenge
49 int i = numsSize - 1;
50 for (;;)
51 {
52     i > 1;
53     cout << nums[i] << "\n";
54     i--;
55     if(i == 1)
56     {
57         break;
58     }
59 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
600
500
400
300
```



## #050 - Loop With For - Nested Loop

اولاً loop : هو loop بداخله آخر مثل الـ Nested loop

```

9 int main ()
10 {
11     string products[] = {"Item 1", "Item 2", "Item 3"};
12     string sizes[] = {"Small", "Large", "X-Large"};
13
14     for (int i = 0; i < 3; i++)
15     {
16         cout << "Product Name:\n";
17         cout << products[i] << "\n";
18         cout << "Sizes:\n";
19         for (int j = 0; j < 3; j++)
20         {
21             cout << sizes[j];
22             if (j < 2)
23             {
24                 cout << ", ";
25             }
26         }
27         cout << "\n";
28         cout << "===== \n";
29     }

```

```

[Running] cd "f:\Programming\"
++\#050 - Loop With For - N
Web School\#2 - Fundamental
Product Name:
Item 1
Sizes:
Small, Large, X-Large
=====
Product Name:
Item 2
Sizes:
Small, Large, X-Large
=====
Product Name:
Item 3
Sizes:
Small, Large, X-Large
=====
[Done] exited with code=0 in 0.013s

```

## #051 - Loop With While

While الـ Syntax -

```

while (Condition Is True)
{
}

```

- مثال: يدل على أن الـ while تشبه الـ loop لكن لها استخدامات مختلفة

```

15 int main()
16 {
17     for (int i = 0; i < 5; i++)
18     {
19         cout << i << "\n";
20     }
21
22     int i = 0;
23
24     while (i < 5)
25     {
26         cout << i << "\n";
27         i++;
28     }

```

```

0
1
2
3
4

```



- ونستطيع استخدام if داخل while وان نقوم بقطع شرط ال while

- وبتغيير الترتيب تتغير النتيجة في ال Terminal

```
22 int i = 0;
23
24 while (i < 5)
25 {
26     cout << i << "\n";
27     i++;
28
29     if (i == 2)
30     {
31         break;
32     }
33 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

0  
1

```
22 int i = 0;
23
24 while (i < 5)
25 {
26     cout << i << "\n";
27
28     if (i == 2)
29     {
30         break;
31     }
32     i++;
33 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

0  
1  
2

## #052 - Loop With Do While

١ - ال Do While : هو أن تقوم بتنفيذ ال Block of code أولًا ثم التأكد من صحة الشرط

عكس ال Condition while ال .Block of code أولًا ثم تنفيذ ال

Do While ال Syntax -

```
do
{
}
} while (Condition is True)
```



## مثالين لل while

```

19 int index = 4;
20
21 while (index < 6)
22 {
23     cout << index << "\n";
24     index++;
25 }

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

4  
5

```

19 int index = 6;
20
21 while (index < 6)
22 {
23     cout << index << "\n";
24     index++;
25 }

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

[Running] cd "f:\Programming\4- Elzero Web School"
"f:\Programming\4- Elzero Web Sch"
[Done] exited with code=0 in 0.32

## مثالين لل do while

```

24 int index = 4;
25
26 do
27 {
28     cout << index << "\n";
29     index++;
30 } while (index < 6);

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

4  
5

```

24 int index = 6;
25
26 do
27 {
28     cout << index << "\n";
29     index++;
30 } while (index < 6);

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

6

## #053 - Loop - Break, Continue

for : هو أن تقوم بعمل skip داخل ال Array عند عمل Continue - ١

Array لـ loop

```

int nums[] = {10, 20, 30, 40, 50};

for (int i = 0; i < 5; i++)
{
    if (nums[i] == 20)
    {
        continue; // Skip Current Iteration And Continue
    }

    cout << nums[i] << "\n";
    cout << "After\n";
}

```

Of Prog	Continue
10	After
30	After
40	After
50	After
	[Done]

ولكن إذا قمنا بعمل ال skip بعد ال cout لن يحدث شيء لأن الفعل سوف يكون حدث وقام بطباعتها بالفعل قبل ان يقوم بعمل ال skip لها.

```

int nums[] = {10, 20, 30, 40, 50};

for (int i = 0; i < 5; i++)
{
    cout << nums[i] << "\n";
    cout << "After\n";

    if (nums[i] == 20)
    {
        continue; // Skip Current Iteration And Continue
    }
}

```

Contin	10
10	After
20	After
30	After
40	After
50	After
	[Done]



والدليل إذا قمنا بوضع cout After continue سيقوم بطباعة رقم 20 ولن يقوم بطباعة الـ After لأنه قام بعمل skip للـ iteration وهي الـ After

```
int nums[] = {10, 20, 30, 40, 50};

for (int i = 0; i < 5; i++)
{
    cout << nums[i] << "\n";

    if (nums[i] == 20)
    {
        continue; // Skip Current Iteration And Continue
    }

    cout << "After\n";
}
```

"f:\Пр  
оф Пр  
Conti  
10  
After  
20  
30  
After  
40  
After  
50  
After

**ملاحظة:** عند وجود رقمين 20 في الـ Array سوف يقوم بعمل له skip أيضاً لأنه عبارة عن الـ iterations بين الـ Array الموجودين الموجدين في الـ iteration

```
int nums[] = {10, 20, 30, 40, 20, 50};

for (int i = 0; i < 6; i++)
{
    if (nums[i] == 20)
    {
        continue; // Skip Current Iteration And Continue
    }
    cout << nums[i] << "\n";
    // cout << "After\n";
}
```

Fu  
Bre  
"f:  
Of  
Cor  
10  
30  
40  
50  
[Do

**iteration - ٢:** هو ان يقوم بالوقوف عند element معين في المصفوفة وتوقف الـ break عند هذا الـ element

```
int main()
{
    int nums[] = {10, 20, 30, 40, 20, 50};

    for (int i = 0; i < 6; i++)
    {
        if (nums[i] == 10)
        {
            break;
        }
        cout << nums[i] << "\n";
    }
}
```

[Done]  
[Run]  
Fundame  
Break,  
"f:\Пр  
оф Пр  
Conti  
10  
[Done]

وإذا كان الـ break بعد الـ cout وليس قبلها سوف يقوم بطباعة الـ element ثم يقوم بوقف الـ iteration من بعده

```
int nums[] = {10, 20, 30, 40, 20, 50};

for (int i = 0; i < 6; i++)
{
    cout << nums[i] << "\n";

    if (nums[i] == 10)
    {
        break;
    }
}
```

[Dor  
[Run  
Fundame  
Break,  
"f:\Пр  
оф Пр  
Conti  
10  
[Dor



## #054 - Loop Training Create Three Apps

### Loop

- Compare

--- For => Specific Number Of Loops

--- While => Loop As Long Condition Is True

--- Do While => Always Execute Once

### Create Three Apps

--- Count Positive & Even Numbers Only

--- Guess The Number

--- Reversed Elements From User

١ - **For**: تستخدم عندما يوجد لديك عدد معين من ال iterations تعرفه أي معروف عددهم

٢ - **While**: عندما يكون لدينا شرط اثناء ما الشرط صحيح أي اجابته true سوف يستمر في ال loop حتى يكون الشرط خطأ false

٣ - **Do While**: مثل ال While لكن تنفذ شيء في البداية

### - Create Three Apps

#### 1- Count Positive & Even Numbers Only

```
17 int main()
18 {
19     // Count Positive & Even Numbers Only
20     int result = 0;
21     int nums[] = {10, 20, -20, 13, 30, -30, 40};
22     int numsSize = size(nums); // 7
23
24     for(int i = 0; i < numsSize; i++)
25     {
26         if (nums[i] > 0 && nums[i] % 2 == 0)
27         {
28             result += nums[i];
29         }
30     }
31
32     cout << "Result Is: " << result;

```

Result Is: 110  
[Done] exited with code 0

[Running] cd "f:\Programming\CPP\#054 - Loop Training"  
With C++\#054 - Loop Training  
Result Is: 100  
[Done] exited with code 0

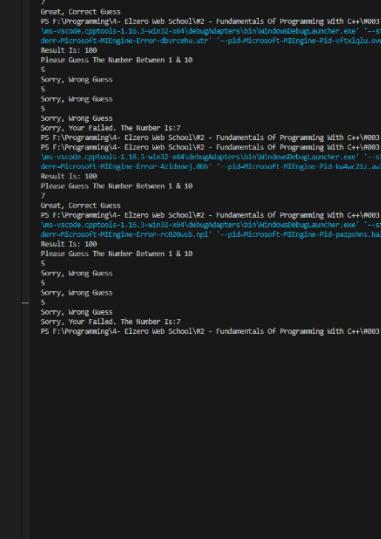
[Running] cd "f:\Programming\CPP\#054 - Loop Training"  
With C++\#054 - Loop Training  
Result Is: 100  
[Done] exited with code 0



## 2- Guess The Number

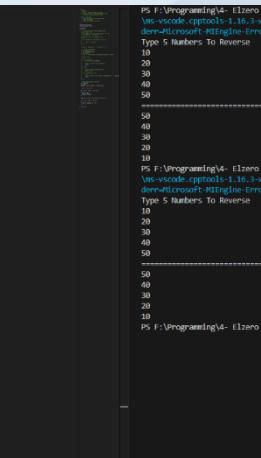
```
// Guess The Number
int guessNumber = 7;
int guessTries = 0;
int choose;
cout << "Please Guess The Number Between 1 & 10\n";

while (true)
{
    cin >> choose;
    if (choose == guessNumber)
    {
        cout << "Great, Correct Guess\n";
        break;
    }
    else
    {
        cout << "Sorry, Wrong Guess\n";
        guessTries++;
    }
    if (guessTries == 3)
    {
        cout << "Sorry, You Failed. The Number Is:" << guessNumber;
        break;
    }
}
```



## 3- Reversed Elements From User

```
-- 60 // Reversed Elements From User
61 int vals[5];
62 int inp;
63 cout << "Type 5 Numbers To Reverse\n";
64
65 // [0, 1, 2, 3, 4]
66
67 for (int i = 4; i > -1; i--)
68 {
69     cin >> inp;
70     vals[i] = inp;
71 }
72
73 cout << "===== \n";
74
75 for (int i = 0; i < 5; i++)
76 {
77     cout << vals[i] << "\n";
78 }
```



## #055 - Function - Introduction

### Function

- DRY - Don't Repeat Yourself
- User Defined and Built-In
- Syntax
- Example
- Why We Use Functions
- Declare A Function And Call It



هي **Function - Block of Code** أي مجموعة من الأسطر البرمجية تنفذ بها مهمة معينة (Task) أو أكثر من مهمة

١- هي تطبيق لمبدأ **DRY – Don't Repeat Yourself**

٢- يوجد نوعين من الـ **Function** هما

١. **Standard Function** : وتسمى **Built-In Function** أيضاً وهي موجودة في اللغة جاهزة ونقوم باستخدامها مباشرة.

٢. **User Defined Function** : هي الـ **Function** التي نقوم بإنشائها ونقوم بتسميتها باسم الذي نريده ونقوم من خلالها بإرجاع نوع البيانات الذي نريده ونجعلها تقوم بتنفيذ المهمة الذي نريد تنفيذها.

## Syntax -٣

```
returnDataType functionname(Param1, Param2, Param3)
{
    // Function Body Contain Block Of Code
}
```

**ملاحظة:** الـ **Parameter** هو معامل التجربة

## Declare A Function And Call It -٤

```
20 // Declare Function
21 void sayHello()
22 {
23     cout << "Hello Osama.\n";
24 }
25
26 int main()
27 {
28     cout << "Hello Ahmed.\n";
29     sayHello();
30     cout << "Hello Sayed.\n";
31     return 0;
32 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Hello Ahmed.  
Hello Osama.  
Hello Sayed.

**ملاحظة:** **void** معناها فراغ أو فارغ



## #056 - Function With Parameter

- ال parameter يوضع بين ال square brackets وعنده استدعاء ال function نقوم بذكر اسم ال parameter في ال square brackets

```
int main() void sayHello(std::string
{
    name)
    sayHello()
```

**ملاحظة:** الذي يكتب في ال parameter يسمى function ولكن الذي يكتب عند عمل call/invoke حيث يعتبر ال parameter يسمى argument هو ال variable هو ال value وال argument هو ال variable

```
14 void sayHello(string name)
15 {
16     cout << "Hi " << name << ".\n";
17 }
18
19 int main()
20 {
21     sayHello("Fady");
22     cout << "Hi Ahmed.\n";
23     cout << "Hi Sayed.\n";
24     return 0;
25 }
```

**ملاحظة ٢:** عند وضع 2 parameters أو أكثر يجب أن تكتب كل قيمة من قيم ال arguments في ال parameters من نفس النوع.

```
9 void sayHello(string msg, string name)
10 {
11     cout << msg << " " << name << ".\n";
12 }
13 could not convert '123' from 'int' to 'std::string' {aka 'std::__cxx11::basic_string<char>'} gcc
14 int main()
15 {
16     sayHello(123, "Fady");
```

**ملاحظة ٣:** يجب أن يكون عدد ال arguments عند استدعاء ال function نفس عدد المكتوبين في ال parameters

```
9 void sayHello(string msg, string name)
10 {
11     cout << msg << " " << name << ".\n";
12 }
13 too few arguments to function 'void sayHello(std::string, std::string)' gcc
14 in void sayHello(std::string msg, std::string name)
15 {
16     sayHello("Fady"); // Problem
```

```
9 void sayHello(string msg, string name)
10 {
11     cout << msg << " " << name << ".\n";
12 }
13 too many arguments to function 'void sayHello(std::string, std::string)' gcc
14 in void sayHello(std::string msg, std::string name)
15 {
16     sayHello("Fady", "Ahmed", "Sayed"); // Problem
```



## #057 - Function With Parameter Training

مثال ١ :

```

11 void iceBox(string item)
12 {
13     if (item == "Coca Cola")
14     {
15         cout << item << " Will Be More Cold\n";
16     }
17     else if (item == "Apple" || item == "Juice")
18     {
19         cout << item << " Will Be More Fresh\n";
20     }
21     else
22     {
23         cout << item << " Is Invalid\n";
24     }
25 }
26
27 int main()
28 {
29     iceBox("Coca Cola");
30     iceBox("Apple");
31     iceBox("Juice");
32     iceBox("TV Remote");
33     return 0;
34 }

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Coca Cola Will Be More Cold
Apple Will Be More Fresh
Juice Will Be More Fresh
TV Remote Is Invalid

```

```

27 void iceBox(string item, string event)
28 {
29     if (item == "Coca Cola")
30     {
31         cout << item << " " << event;
32     }
33     else if (item == "Apple" || item == "Juice")
34     {
35         cout << item << " " << event;
36     }
37     else
38     {
39         cout << item << " " << event;
40     }
41 }
42
43 int main()
44 {
45     iceBox("Coca Cola", "Will Be More Cold\n");
46     iceBox("Apple", "Will Be More Fresh\n");
47     iceBox("Juice", "Will Be More Fresh\n");
48     iceBox("TV Remote", "Is Invalid");
49     return 0;
50 }

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Coca Cola Will Be More Cold
Apple Will Be More Fresh
Juice Will Be More Fresh
TV Remote Is Invalid

```

الذي على اليسار parameter واحد ولكن الذي على اليمين بواسطة 2 parameter

مثال ٢ :

```

43 void calc(int numOne, int numTwo, string op)
44 {
45     if (op == "+")
46     {
47         cout << numOne << " + " << numTwo << " = ";
48         cout << numOne + numTwo << "\n";
49     }
50     else if (op == "-")
51     {
52         cout << numOne << " - " << numTwo << " = ";
53         cout << numOne - numTwo << "\n";
54     }
55     else if (op == "*")
56     {
57         cout << numOne << " * " << numTwo << " = ";
58         cout << numOne * numTwo << "\n";
59     }
60     else if (op == "/")
61     {
62         cout << numOne << " / " << numTwo << " = ";
63         cout << numOne / numTwo << "\n";
64     }
65 }

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```

```

67 int main()
68 {
69     calc(10, 20, "+");
70     calc(40, 20, "-");
71     calc(5, 10, "*");
72     calc(50, 10, "/");
73     return 0;
74 }

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
10 + 20 = 30
40 - 20 = 20
5 * 10 = 50
50 / 10 = 5

```



## #058 - Function Parameter Default Value

Parameter Default Value - هو القيمة الافتراضية لـ parameter أي عندما لا يوجد قيمة لـ parameter عند استدعاء الـ function يأخذ القيمة الافتراضية

```
14 void details(string msg = "Welcome", string name = "Unknown")
15 {
16     cout << msg << " " << name << "\n";
17 }
18
19 int main()
20 {
21     details("Hello", "Ahmed");
22     details("Hi");
23     details();
24     return 0;
25 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
Hello Ahmed
Hi Unknown
Welcome Unknown
```

ملحوظة ١: عند وضع parameter الثاني لن يحدث error لـ Default Value

```
14 void details(string msg, string name = "Unknown")
15 {
16     cout << msg << " " << name << "\n";
17 }
18
19 int main()
20 {
21     details("Hello", "Ahmed");
22     details("Hi");
23     // details();
24     return 0;
25 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
Hello Ahmed
Hi Unknown
```

ملحوظة ٢: عند وضع الأول وعدم وضع parameter لـ Default Value لن يحدث error لـ parameter second Value

```
10
11
12
13
14 void details(string msg = "Welcome", string name)
15 {
16     cout << msg << " " << name << "\n";
17 }
18
19 int main()
20 {
21     details("Hello", "Ahmed");
22     details("Hi");
23     // details();
24     return 0;
25 }
```

default argument missing for parameter 2 of 'void details(std::string, std::string)' gcc  
std::string name



## #059 - Passing Array As Parameter

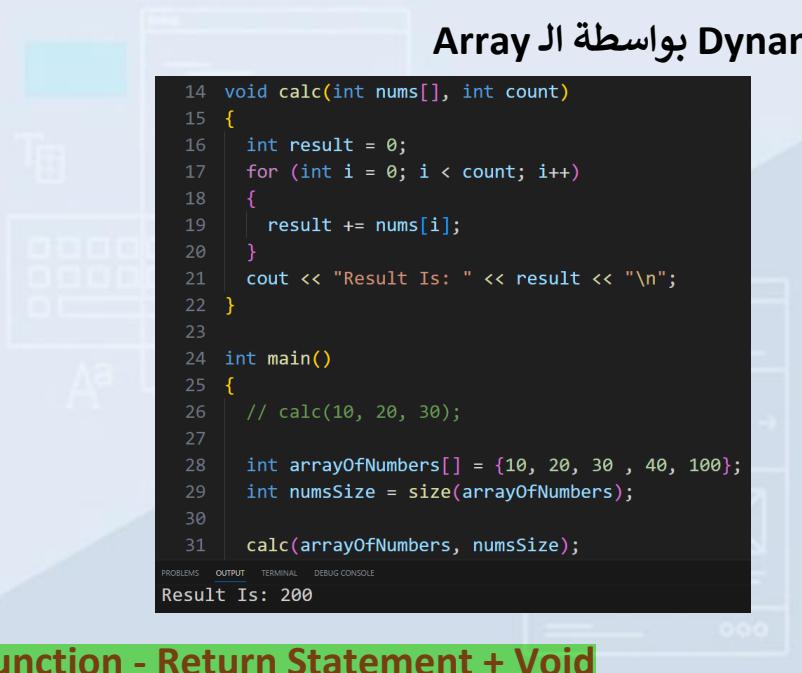
حيث يقوم بالتعديل والجمع عند زيادة كل رقم هكذا لذا نستخدم الـ **Array** لكي نجعل الـ **Dynamic Function**

```

16 void calc(int n1, int n2, int n3)
17 {
18     cout << n1 + n2 + n3 << "\n";
19 }
20
21 int main()
22 {
23     calc(10, 20, 30);
24     return 0;
25 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE  
60

## الـ **Array** بواسطة الـ **Dynamic Function** -



```

14 void calc(int nums[], int count)
15 {
16     int result = 0;
17     for (int i = 0; i < count; i++)
18     {
19         result += nums[i];
20     }
21     cout << "Result Is: " << result << "\n";
22 }
23
24 int main()
25 {
26     // calc(10, 20, 30);
27
28     int arrayOfNumbers[] = {10, 20, 30, 40, 100};
29     int numsSize = size(arrayOfNumbers);
30
31     calc(arrayOfNumbers, numsSize);
}
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE  
Result Is: 200

## #060 - Function - Return Statement + Void

يقوم بعمل **action** معين في الـ **void system** فقط وهي الـ **No Return Function - ١ function**

مثل هذا المثال: لا يقوم بإرجاع قيمة، ولكن يقوم بطبع القيمة في الـ **Terminal** مباشرة

```

16 void calc(int n1, int n2)
17 {
18     cout << n1 + n2 << "\n";
19 }
20
21 int main()
22 {
23     calc(10, 20);
24     return 0;
25 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE  
30



```
18 #include <iostream>
19 using namespace std;
20
21 void calc(int n1, int n2)
22 {
23     cout << n1 + n2 << "\n";
24 }
25
26 int main()
27 {
28     calc(10, 20);
29     int result = calc(10, 20);
30     return 0;
31 }
```

100 % Error List Build + IntelliSense

Entire Solution 2 Errors 0 Warnings 0 Messages

E0144 a value of type "void" cannot be used to initialize an entity of type "int"  
C2440 'initializing': cannot convert from 'void' to 'int'

**ملاحظة:** لا يمكن ان نقوم بعمل assign لقيمة الـ **variable** داخل void function (a value of type "void" cannot be used to initialize an entity of type "int")

حيث هنا نقوم بعمل متغير int ونريد أن نضع بداخله قيمة وهذه القيمة لم تقوم بالاسترجاع return

**٢ - Return Function:** مثل الآلة الحاسبة نقوم بكتابة رقم ونجمع عليه رقم آخر ثم نضغط = يقوم بإعادة قيمة لنا ومن الممكن أن نقوم بأخذ هذه القيمة وعمل له عملية أخرى حيث يوجد قيمة حقيقة نتعامل معها.

مثال: على اليسار عند استدعاء الـ cout في الـ terminal ينتج الـ function دون تأثير وعلى اليمين الـ function تطبع قيمة ولكنها تقوم بإرجاع قيمة تقوم باستخدامها كما تشاء

```
21 int calc(int n1, int n2)
22 {
23     cout << "Operation Is Done\n";
24     return n1 + n2;
25 }
26
27 int main()
28 {
29     calc(10, 20);
30     // int result = calc(10, 20);
31     // cout << result * 5 << "\n";
32     // cout << result + 20 << "\n";
33     return 0;
34 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Operation Is Done

```
21 int calc(int n1, int n2)
22 {
23     cout << "Operation Is Done\n";
24     return n1 + n2;
25 }
26
27 int main()
28 {
29     // calc(10, 20);
30     int result = calc(10, 20);
31     cout << result * 5 << "\n";
32     cout << result + 20 << "\n";
33     return 0;
34 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Operation Is Done  
150  
50



**ملحوظة:** الـ **main function** يجب ان يكون نوعها **int** ولا ينفع أن يكون **void**

```
25 ' ::main' must return 'int' gcc
26 View Problem [Alt+F8] No quick fixes available
27 void main()
28 {
```

أي عملية او طباعة بعد الـ **return** لن تنفذ حيث أن الـ **function** عندما يصل إلى الـ **return** يقوم بالعودة لأعلى ليرجع شيء من الـ **compiler** ولا يكمل بعد الـ **return**

```
21 int calc(int n1, int n2)
22 {
23     cout << "Operation Is Done\n";
24     return n1 + n2;
25     cout << "Will Not Show";
26 }
27
28 int main()
29 {
30     calc(10, 20);
}
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Operation Is Done
```

**Void With Return -٤**: لا يمكن ان نقوم بإرجاع قيمة **integer** داخل **void** حيث ان الـ **function** عدد والـ **function** فارغة.

```
16 void calc(int n1, int n2)
17 {
18     cout << View Problem [Alt+F8] No quick fixes available
19     return 10;
20 }
```

لكن عند وضع **return;** فقط لن يحدث **error** لأنها تعتبر **void** أي فارغة ونقوم بعمل الـ **return;** لأنها تقوم بعمل الـ **break;** أي قم بانهاء الـ **function** هنا

```
21 void calc(int n1, int n2)
22 {
23     cout << n1 + n2 << "\n";
24     return;
25 }
26
27 int main()
28 {
29     calc(10, 20);
}
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
30
```



## #061 - Function - Forward Declaration

١- لا يجب عمل function call/invoke قبل الإعلان - Declaration عن function

```
/ using namespace std;
8   'calc' was not declared in this scope; did you mean 'calloc'? gcc
9 int main() int calc(int a, int b)
10 {
11   // Call/ Declaration
12   cout << calc(10, 20);
13   return 0;
14 }
15
16 // Declaration
17 int calc(int a, int b)
18 {
19   return a + b;
20 }
```

أي التسبيق بالإعلان عن الـ function قبل أن يتم عمل Forward Declaration -٢ من الأساس ومن الممكن تكون في نفس الملف، ولكن في الأسفل أو في ملف آخر



```
9 #include <iostream>
10 using namespace std;
11
12 int calc(int a, int b);
13
14 int main()
15 {
16   // Call/Invoke
17   cout << calc(10, 20);
18   return 0;
19 }
20
21 // Declaration
22 int calc(int a, int b)
23 {
24   return a + b;
25 }
```

```
9 #include <iostream>
10 using namespace std;
11
12 int calc(int a, int b);
13
14 // Declaration
15 int calc(int a, int b)
16 {
17   return a + b;
18 }
19
20 int main()
21 {
22   // Call/Invoke
23   cout << calc(10, 20);
24   return 0;
25 }
```

30

30

المثال الذي على اليسار لتصحيح الخطأ

لكن الذي على اليمين لتوضيح أن الـ Forward Declaration لن يؤثر إذا كان الـ function call استدعاء الـ function declare



**ملحوظة:** لكن لا يمكن الإعلان عن الـ **function** مرتين قبل أن نقوم بعمل **redefinition error** وهو خطأ آخر يحدث بعده سوف.

```
9 // Declaration
10 int calc(int a, int b)
11 {
12     return a + b;
13 }
14
15 int main()
16 {
17     // Call/Invoke
18     cout << calc(10, 20);
19     redefinition of 'int calc(int, int)' gcc
20 } int calc(int a, int b)
21 Declaration
22 // D View Problem (Alt+F8) No quick fixes available
23 int calc(int a, int b)
24 {
25     return a + b;
26 }
```

## #062 - Built-In Functions - Math Functions

- Math Functions

--- pow

--- fmod

--- ceil

--- floor

--- round

--- trunc

power function أي الأس **pow** - ١

```
24 cout << pow(2, 4) << "\n";      // 16
25 cout << 2 * 2 * 2 * 2 << "\n";    // 16
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
16
16
```

modulus function أي باقي القسمة **fmod** - ٢

```
25 cout << fmod(11.5, 2) << "\n"; // 1.5
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
1.5
```



**ملحوظة:** لا يمكن استخدام الـ **modulus operator** مع رقم او رقمين **float** لذا نستخدم **function modulus function**

```
cout << 11 %           invalid operands of types 'double' and 'int' to binary 'operator%' gcc
cout << 11.5 % 2 << "\n"; // Error
```

٣ - **ceil**: بمعنى سقف وهي **function** لتقرير الرقم العشري وتحويله لعدد صحيح ولأنها بمعنى سقف سوف يقوم بتقريرها للرقم الأكبر

```
25 | cout << ceil(9.1) << "\n"; // 10
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
10
```

٤ - **floor**: بمعنى الأرض وهي **function** لتقرير الرقم العشري وتحويله لعدد صحيح ولأنها بمعنى أرض سوف يقوم بتقريرها للرقم الأصغر

```
27 | cout << floor(9.9) << "\n"; // 9
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
9
```

٥ - **round**: يقوم بالتقرير حسب الرقم قريب من الرقم الأكبر أو الأقل حيث يختلف عن الـ **ceil, floor** مهما كانت الكسور سوف يقوموا بالتقرير للأكبر أو الأقل

```
29 | cout << round(9.5) << "\n"; // 10
30 | cout << round(9.4) << "\n"; // 9
31 | cout << round(9.49) << "\n"; // 9
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
10
9
9
```

**ملحوظة:** إذا كان الرقم ٥، أو أعلى من النص يقرب للرقم الأكبر ولكن إذا كان الرقم أقل من النص يقرب للرقم الأصغر

٦ - **trunc**: وهي إزالة الكسور نهائياً وهي بمعنى **truncate** أي اقتطاع

```
33 | cout << trunc(9.9) << "\n"; // 9
34 | cout << trunc(9.5) << "\n"; // 9
35 | cout << trunc(9.1) << "\n"; // 9
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
9
9
9
```



## #063 - Built-In Functions - Training - Create 2 Apps

### Function

#### - Built-In Functions

##### --- cctype Functions

----- `tolower()`

----- `toupper()`

----- `isupper()`

----- `islower()`

----- `isspace()`

#### - Create 2 Applications

##### --- Swap Case App

##### --- Remove Spaces App

. ١ : لجعل الحرف small ascii value ولكن ينتج قيمة الـ `tolower()`

```
22 cout << "A\n"; // A
23 cout << tolower('A') << "\n"; // 97 => ASCII Value
24 cout << char(tolower('A')) << "\n"; // a
25 cout << char(97) << "\n"; // a
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

A  
97  
a  
a

. ٢ : لجعل الحرف capital ascii value ولكن ينتج قيمة الـ `toupper()`

```
27 cout << "b\n"; // b
28 cout << toupper('b') << "\n"; // 66 => ASCII Value
29 cout << char(toupper('b')) << "\n"; // B
30 cout << char(66) << "\n"; // B
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

b  
66  
B  
B



## السؤال ٣ - isupper(): إذا كان الحرف Boolean Expression

السؤال ٤ - islower(): إذا كان الحرف small

**مثال Swap Case App:**

```
34 // Swap Case App
35 string nameone = "ElZZero"; // eLzeRO eLzeRO
36 int nameoneSize = size(nameone);
37
38 for (int i = 0; i < nameoneSize; i++)
39 {
40     if (isupper(nameone[i]))
41     {
42         cout << char(tolower(nameone[i]));
43     }
44     else
45     {
46         cout << char(toupper(nameone[i])));
47     }
48
49 // cout << nameone[i] << "\n";
50 // cout << int(nameone[i]) << "\n";
51 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

eLzeRO

السؤال ٥ - isspace(): لإزالة جميع المسافات بما بينهم مسافة السطر الجديد وال tab

```
54 // Remove Spaces App
55 string nametwo = "E\nl z \n\te r\t\no";
56 int nametwoSize = size(nametwo);
57
58 for (int i = 0; i < nametwoSize; i++)
59 {
60     // if (nametwo[i] == ' ')
61     // {
62     //     continue;
63     // }
64     if (isspace(nametwo[i]))
65     {
66         continue;
67     }
68     cout << nametwo[i];
69 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Elzero



## #064 - Built-In Functions - Training - Create 3 Apps

### Function

#### - Built-In Functions

##### --- Algorithm Header

----- min

----- max

----- count

#### - Create 3 Applications

##### --- Find Minimum Number

##### --- Find Maximum Number

##### --- Count Number Occurance

١ - **min**: للمقارنة بين رقمين Contain أو مجموعة من الأرقام Array of Numbers أو الحروف واستخراج الرقم الأصغر أو الحرف الذي له أقل ASCII Value

```
cout << min(10, -20) << "\n"; // -20
cout << min(10, 20) << "\n"; // 10
cout << min('a', 'c') << "\n"; // a
cout << min('a', 'C') << "\n"; // c
cout << int('a') << "\n"; // 97
cout << int('c') << "\n"; // 99
cout << int('C') << "\n"; // 67
cout << min({10, -20, 30, -100, 100, -50}) << "\n"; // -100
```

```
-20
10
a
c
97
99
67
-100
```

٢ - **max**: للمقارنة بين رقمين Contain أو مجموعة من الأرقام Array of Numbers أو الحروف واستخراج الرقم الأكبر أو الحرف الذي له أكبر ASCII Value

```
cout << max(10, -20) << "\n"; // 10
cout << max(10, 20) << "\n"; // 20
cout << max('a', 'c') << "\n"; // c
cout << max('a', 'C') << "\n"; // a
cout << int('a') << "\n"; // 97
cout << int('c') << "\n"; // 99
cout << int('C') << "\n"; // 67
cout << max({10, -20, 30, -100, 100, -50}) << "\n"; // 100
```

```
10
20
c
a
97
99
67
100
```



## Find Minimum Number App -₹

```
40 // Find Minimum Number App
41 int nums[] = {10, -20, 30, -100, 100, -50};
42 int numsSize = size(nums);
43 int checkMinNum = 0;
44
45 for (int i = 0; i < numsSize; i++)
46 {
47     if (nums[i] < checkMinNum)
48     {
49         checkMinNum = nums[i];
50     }
51 }
52
53 cout << "Minimum Number Is " << checkMinNum << "\n";
```

PROBLEMS    **OUTPUT**    TERMINAL    DEBUG CONSOLE

-100

## Find Maximum Number App -₹

```
55 // Find Maximum Number App
56 // int nums[] = {10, -20, 30, -100, 100, -50};
57 // int numsSize = size(nums);
58 int checkMaxNum = 0;
59
60 for (int i = 0; i < numsSize; i++)
61 {
62     if (nums[i] > checkMaxNum)
63     {
64         checkMaxNum = nums[i];
65     }
66 }
67
68 cout << "Maximum Number Is " << checkMaxNum << "\n";
```

PROBLEMS    **OUTPUT**    TERMINAL    DEBUG CONSOLE

Maximum Number Is 100

## Count Number Occurrence App -₹

```
76 // Count Number Occurance App
77 int numsTwo[] = {10, 20, 10, 10, 13, 15, 100, 20, 10};
78 int numsTwoSize = size(numsTwo);
79 int counter = 0;
80 int choosenNum = 10;
81
82 for (int i = 0; i < numsTwoSize; i++)
83 {
84     if (numsTwo[i] == choosenNum)
85     {
86         counter++;
87     }
88 }
89
90 cout << choosenNum << " Found " << counter << " Times";
```

PROBLEMS    **OUTPUT**    TERMINAL    DEBUG CONSOLE

10 Found 4 Times



## #065 - Function Overloading

Function Overloading - ١ : وهو أن تقوم بعمل أكثر من function بنفس الأسم ولكن بإختلاف الـ parameters أو الـ types أو الاثنين معاً.

```

14 void print(int a, int b)
15 {
16     cout << "Number One Is: " << a << "\n";
17     cout << "Number Two Is: " << b << "\n";
18 }
19
20 void print(int a, int b, int c)
21 {
22     cout << "Number One Is: " << a << "\n";
23     cout << "Number Two Is: " << b << "\n";
24 }

PROBLEMS 1 OUTPUT TERMINAL DEBUG CONSOLE
Number One Is: 10
Number Two Is: 20

```

لن يحدث error لأن هذه الـ function الأخرى تسمى overloaded function لأنها تمت بـ function overloading لكن عدد الـ parameters مختلف

**ملاحظة:** يحدث error عندما يكونوا نفس عدد الـ parameters ويخبرك بأن قومت function لـ redeclare بـ function overloading

```

14 void print(int a, int b)
15 {
16     cout << "Number One Is: " << a << "\n";
17     cout redefinition of 'void print(int, int)' gcc
18 }     void print(int a, int b)
19
20 void print(int a, int b)
21 {
22     cout << "Number One Is: " << a << "\n";
23     cout << "Number Two Is: " << b << "\n";
24 }

View Problem (Alt+F8) Quick Fix... (Ctrl+.)

```

**ملاحظة ٢:** وعند استدعاء الـ function الأولى وعند وضع عدد الثانية يقوم باستدعائهما function باـ parameters

```

int main()
    1/2 void print(int a, int b)
print(10, 20);
print(100, 200)

→ int main()
    void print(int a, int b,
    print(10, 20); 2/2 int c)
    print(100, 200, )

```



parameters type مع تغير الـ function overloading - ٢

```
23 void print(string a, string b)
24 {
25     cout << "Text One Is: " << a << "\n";
26     cout << "Text Two Is: " << b << "\n";
27 }
28
29 int main()
30 {
31     print(10, 20);
32     print(100, 200, 300);
33     print("Fady", "Alamir");

```

```
23 void print(string a, int b)
24 {
25     cout << "Text One Is: " << a << "\n";
26     cout << "Text Two Is: " << b << "\n";
27 }
28
29 int main()
30 {
31     print(10, 20);
32     print(100, 200, 300);
33     print("Fady", 10);
```

## #066 - Function Recursion

Function Recursion - هو عبارة عن technique من خلاله نجعل الا ت function نفسها call/invoke

```
11 int add(int num)
12 {
13     if (num == 0)
14     {
15         return 0;
16     }
17     cout << num << "\n";
18     cout << "=====\n";
19     return num + add(num - 1);
20 }
21
22 // 5 + (add(4))
23 // 5 + ( 4 + add(3) )
24 // 5 + ( 4 + ( 3 + add(2) ) )
25 // 5 + ( 4 + ( 3 + ( 2 + add(1) ) ) )
26 // 5 + ( 4 + ( 3 + ( 2 + ( 1 + add(0) ) ) ) )
27
28 int main()
29 {
30     cout << add(5);
31     return 0;
32 }
```

15  
[Done] exited with c

[Running] cd "f:\Pro  
Function Recursion\"  
Of Programming With  
5  
=====

4  
=====

3  
=====

2  
=====

1  
=====

15  
[Done] exited with c

[Running] cd "f:\Pro  
Function Recursion\"  
Of Programming With



## #067 - Vector - What Is Vector

```
/*
Vector
- What Is Vector ?
--- Vector Is A Container For Similar Data Like Array
--- Vectors Are Dynamic Arrays => Array That Can Change In Size
--- Vector Is A Class Template
- Vector Syntax => vector<type> VariableName
- Vector Create With All Methods
- Loop On Elements
- Important Notes

We Will Cover The Comparison With Array Later
*/
```

١- عبارة عن **Vector** أو حاوية نصيف بداخلها مجموعة من البيانات المتشابهة، مجموعة من الأرقام أو الـ **characters** أو الـ **strings** وهكذا..

٢- الـ **Vector** عبارة عن **Dynamic Array** أي نستطيع عمل **Resize** لها بعد انشاءها أي نستطيع ان نقوم بزيادة او نقص عنصر على عكس الـ **Array** بعد انشاءها تكون **Fixed** لا نستطيع عمل **Resize** لها

٣- الـ **Vector** عبارة عن **Class** وهي مخصصة مع الـ **OOP**

٤- الـ **vector<type> VariableName** ← **Vector Syntax**

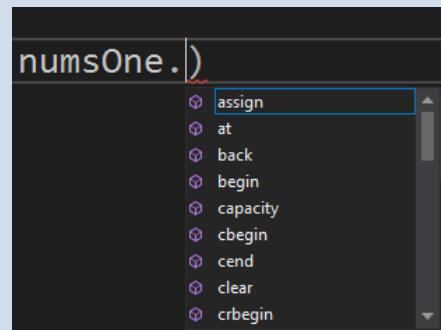
**ملاحظة:** في هذا المثال `vector<int> numsThree(4, 50);` قمنا بتحديد الـ **size** لكن نستطيع ايضاً ان نقوم بعمل **Resize** لها حيث أن الـ 4 هي الـ **size** أي ٤ عناصر والا 50 أول قيمة في الـ **Vector**

يحدث **error** إذا قمنا باضافته للـ **[4]**

```
numsThree[4] = 1000; ✘
```

Expression: vector subscript out of range

**ملاحظة ٢:** تظهر خصائص الـ **class** بعد الـ **dot**





## Loop on Elements -٥

```
int main()
{
    vector<int> numsOne = { 10, 20, 30, 40 };
    vector<int> numsTwo{ 100, 200, 300, 400 };
    vector<int> numsThree(4, 50);

    for (int i = 0; i < numsOne.size(); i++)
    {
        cout << numsOne.at(i) << " ";
    }

    cout << "\n=====\\n";

    return 0;
}
```

**ملاحظة ٣:** يمكن تعديل عنصر في ال `vector` التي لا نستطيع التعديل عليها التي بـ `at` بواسطة parentheses هكذا

واضافة عنصر اضافي بواسطة خاصية ال `class` وهي ال `push_back`

```
vector<int> numsThree(4, 50);

48    numsThree.push_back(1000);
49
50    cout << "Numbers of Elements Is: " << numsThree.size() << "\\n";
51
52    cout << "\\n=====\\n";
53
54    numsThree.at(0) = 1000;
55
56    for (int i = 0; i < numsThree.size(); i++)
57    {
58        cout << numsThree.at(i) << " ";
59    }
60
61    cout << "\\n=====\\n";
```

PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE

Numbers of Elements Is: 5

=====

1000 50 50 50 1000

=====



## #067 - Vector Versus Array

```
-- Vector
--- It Need A Standard Header To Work
--- Can Be Resized After Insertion Or Deletion Of Elements
--- Not Index Based And Elements Accessed By Iterations
--- Vectors Are Slower Than Arrays
--- Vectors Are Occupy More Memory
--- Vector Available In C++ Only

-- Array
--- C-Array Is Language Construct
--- Cannot Be Resized After Its Defined
--- Elements Accessed By Indexes
--- Arrays Are Faster Than Vectors
--- Arrays Occupy Less Memory
--- Vector Available In C & C++
```

Vector Versus Array	
Vector	Array
Standard Header File <code>#include &lt;vector&gt;</code>	مكون من مكونات اللغة Language Construct header file لا يحتاج إلى الـ
بعد إنشائه نستطيع إضافة/حذف عنصر لأنه Dynamic ونستطيع أن نقوم بعمل Resize له	لا نستطيع إضافة أو حذف أي عنصر أي لا نستطيع أن نقوم بعملية الـ Resize
العناصر نستطيع أن ذـ access عليها عن طريق الـ iterator فقط لأن عناصر الـ vector not index based	الـ Index Based Array هي الـ index وذـ access على العناصر عن طريق الـ index
الـ Vector لأنـ Dynamic فإن الدخول على العناصر يكون أبطأ من الـ Array	الـ Array أسرع من الـ Vectors
الـ Vector يقوم بحجز مساحة أكبر في الـ Memory من الـ Array	الـ Array تقوم بحجز مساحة أقل في الـ Memory من الـ Vector
الـ Vector موجود في لغة الـ C++ فقط	الـ Array توجد في لغة الـ C & C++ فقط

```
When To Use Vector
--- When We Don't Know The Size Of The List

When We Use Array
--- When It Comes To Performance & Speed
```

**ملحوظة ١:** عندما لا نعرف حجم البيانات الذي سوف نقوم بإضافتها نستخدم الـ **Vector**

مثـلـ: مجموعة من المهارات يقومـ باضافتها المستخدمـ ولا نعرف عددهـا لـذا نـسـتـخـدـمـ الـ **vector** لـوجـودـ سـهـولةـ فيـ إـضـافـةـ الـ بـيـانـاتـ اوـ حـذـفـهاـ

**ملحوظة ٢:** عندما نحتاج إلى السـرـعةـ والأـداءـ نـقـومـ باـسـتـخـدـامـ الـ **Array**

**ملحوظة ٣:** نـسـتـطـيعـ انـ نـنـشـئـ الـ **Dynamic Array** لـكـنـ الـ **Vector** أـفـضـلـ



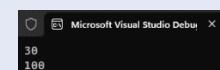
**ملحوظة ٤:** عندما نقوم بإضافة عنصر إلى الـ **Array** يحدث **error** لأن عدد العناصر

محدد

```
int main()
{
    int nums[] = { 10, 20, 30 };
    cout << nums[2] << "\n";
    nums[3] = 100;
    return 0;
}
```

لكن عندما نحدد عدد الـ **elements** داخل الـ **Array** لن يحدث **error**

```
int nums[4] = { 10, 20, 30 };
cout << nums[2] << "\n";
nums[3] = 100;
cout << nums[3] << "\n";
```



**ملحوظة ٥:** وعند إنشاء الـ **Array** بهذه الطريقة أيضًا يحدث **error**

```
array<int, 3> numsArray = { 10, 20, 30 };
cout << numsArray[2] << "\n";
numsArray[3] = 100;
```

Expression: array subscript out of range

وعندما نحدد عدد الـ **elements** داخل الـ **Array** لن يحدث **error**

```
array<int, 4> numsArray = { 10, 20, 30 };
cout << numsArray[2] << "\n";
numsArray[3] = 100;
cout << numsArray[3] << "\n";
```



## Function With Vector Instead of Array -

```
void calc(vector<int> numsVector)
{
    int result = 0;
    for (int i = 0; i < numsVector.size(); i++)
    {
        result += numsVector[i];
    }
    cout << "Result Is: " << result << "\n";
}
```

```
vector<int> arrayOfNumbers = { 10, 20, 30, 40, 100, 300 };
calc(arrayOfNumbers);
```

Result Is: 500



## #069 - Vector - Access, Add, Update And Delete

```
/*
Vector

- Access
--- at()
--- Square Brackets [] < Do Not Use

- Add
--- push_back >>> Add Element To The End

- Update
--- at()

- Delete
--- pop_back() >>> Remove Element From The End
*/
```

١- Access : عندما نريد ان نستدعي element من عناصر ال vector من الأفضل ان نقوم باستخدام `.at()`. لأن ال **square brackets** عندما نقوم باستدعاء عنصر غير موجود في ال **vector** لا نستطيع الاعتماد عليها او تركها في

**التطبيق**

The screenshot shows a code editor interface with a dark theme. On the left, there's a sidebar with icons for file operations. The main area contains the following code:

```
25 → vector<int> nums = { 10, 20, 30 };
26 → // cout << nums.at(3) << "\n";
27 → cout << nums[3] << "\n";
```

Below the code, there are tabs for PROBLEMS, OUTPUT, TERMINAL, and DEBUG CONSOLE. The OUTPUT tab is selected, showing the result of the execution:

```
474
```

لكن ال `.at()` error هو **out of the range**.

The screenshot shows a code editor interface with a dark theme. The code is identical to the previous one, but the output shows an error message:

```
25 → vector<int> nums = { 10, 20, 30 };
26 → cout << nums.at(3) << "\n";
27 → terminate called after throwing an instance of 'std::out_of_range'
what(): std::vector::_M_range_check: __n (which is 3) >= this->size() (which is 3)
```

٢- `push_back` : لاضافة عنصر في نهاية ال **vector**

The screenshot shows a code editor interface with a dark theme. The code adds a new element to the end of the vector:

```
nums.push_back(40);
cout << nums.size() << "\n"; // 4
cout << nums.at(3) << "\n"; // 40
```

To the right, there are two small boxes showing the state of the vector. The first box shows the size as 4 and the fourth element as 40. The second box shows the size as 100 and the fourth element as 100.

٣- `Update` : يمكن تحديث قيمة element في ال **vector** عن طريق ال `.at()`.

The screenshot shows a code editor interface with a dark theme. The code updates the fourth element of the vector:

```
nums.at(3) = 100;
cout << nums.at(3) << "\n"; // 100
```

To the right, there are two small boxes showing the state of the vector. The first box shows the size as 4 and the fourth element as 40. The second box shows the size as 100 and the fourth element as 100.



## ٤ - pop_back(): تقوم بإزالة آخر عنصر من ال vector

```
nums.push_back(500);
cout << nums.size() << "\n"; // 5
cout << nums.at(4) << "\n"; // 500

nums.pop_back();
cout << nums.size() << "\n"; // 4
```

5  
500  
4

## #070 - Vector – Functions

Vector	
- size()	< Current Number Of Elements
- max_size()	< Maximum Number Of Elements
- capacity()	< Storage Capacity
- front()	< First Element
- back()	< Last Element
- clear()	< Clear All Elements From Vector
- empty()	< Check If Its Empty Or No

### ١ - size(): عدد عناصر ال vector الحالي

```
vector<int> nums = { 10, 20, 30, 40 };
cout << nums.size() << "\n"; // 4
```

### ٢ - max_size(): عدد العناصر التي تستطيع أن تقوم بإضافتها في ال vector

```
cout << nums.max_size() << "\n"; 1873741823
```

٣ - capacity(): ال storage الممحوza لعناصر ال vector واحياناً عددها يكون اكثراً من عدد العناصر لأنها تعطينا ال storage وليس عدد العناصر

```
vector<int> nums = { 10, 20, 30, 40 };
nums.push_back(50);
nums.push_back(60);
nums.push_back(70);
nums.push_back(80);
cout << nums.capacity() << "\n";
```

9  
...

### ٤ - front(): تعطي أول عنصر في ال vector

```
cout << nums.front() << "\n"; 10
```

### ٥ - back(): تعطي آخر عنصر في ال vector

```
cout << nums.back() << "\n"; 80
```

**ملاحظة:** ونستطيع عن طريق ال at() استخراج العنصر الأول والأخير أيضاً

```
cout << nums.front() << "\n";
cout << nums.at(0) << "\n";
cout << nums.back() << "\n";
cout << nums.at(nums.size() - 1) << "\n";
```

10  
10  
80  
80



٧ - clear(): من خلالها نقوم بإزالة جميع عناصر ال vector

```
nums.clear();
cout << nums.size() << "\n";
```

٨ - empty(): تستخدم مع ال vector لنقوم بعمل check إذا كان ال vector فارغ أم لا

```
nums.clear();
cout << nums.size() << "\n";
if (nums.empty())
{
    cout << "Vector Is Empty\n";
}
else
{
    cout << "Vector Is Not Empty\n";
}
```

Vector Is Empty

## #071 - Vector - Iterator And Why To Use

```
Vector
- Iterator
--- Containers
----- Array
----- Vector
----- List

--- What Is Iterators
----- Iterators Used To Point To Memory Address Of The Container

--- Why We Use Iterators
[1] Simplify The Code => No Need To See The Full Iteration On Containers
[2] Support For Many Algorithms Like Sorting And Finding
[3] Allow The Dealing With One Element Without The Need To Load The Full List
[4] Work The Same Way With All Containers
[5] It Reduce The Complexity And Execution Time Of The Application

--- Syntax
----- Container<Type>::iterator IteratorName;

--- Initialize
----- With Victor Syntax
----- With Auto Keyword

--- Print
----- [*] Dereference => Don't Print The Iterator, Print What It's Point To

--- Notes
----- This Is Not Pointer, We Will Talk About Pointer Later
```

٩ -Iterator - container : نستخدمه لنشير عنوان الذاكرة Memory Address الخاص بال

- لماذا نستخدم ال Iterator -

١ - يقوم بعمل iterations أي تبسيط للكود ولا نحتاج أن نرى ال Container كاملة الخاصة بال

٢ - يدعم أكثر من Algorithm مثل البحث او عمل sorting والبحث عن البيانات

٣ - نستطيع أن نتعامل مع العنصر من دون نحتاج أن نقوم بعمل load لجميع البيانات

٤ - نستطيع أن نعمل بال containers.Iterator بنفس الطريقة مع كل ال containers



## ٥- يقلل نسبة التعقيد ووقت التنفيذ الخاص بال Application

**(Container<Type>::iterator IteratorName;) :Syntax -**

**مثال:** `vector<int>::iterator it = nums.begin();`

وليس `vector<int> nums = { 10, 20, 30, 40 };` **nums.begin** مجرد طباعة أي من بعد أن نشير له نستطيع أن نقوم بأشياء كثيرة مثل أحضار العنصر الذي بعده بعناصر أو حذف مجموعة العناصر الذي تليه لذا عندما يشير له يكون لل iterator دور فعال.

**- طريقة أخرى لإنشاء ال iterator مباشرة:**

```
auto ite = nums.begin() + 1;
cout << "First Element Is: " << *ite << "\n";
```

**ملحوظة:** عندما نريد طباعة ما يشير إليه ال **Iterator** نضع قبله علامة ال `[*]` هذه النجمة تقوم بعمل شيء يسمى **Dereference** للوصول إلى القيمة أو الكائن الموجود في موقع الذاكرة المخزن في مؤشر.

```
cout << "First Element Is: " << *it << "\n";
cout << "Second Element Is: " << *ite << "\n";
cout << "First Element Is: " << *nums.begin() << "\n";
```

```
First Element Is: 10
Second Element Is: 20
First Element Is: 10
```

**ملحوظة ٢:** وعندنا نعطيه `error` بالعناصر يعطينا `nums.erase(0, 3)` لأنه يريد ال

`nums.erase(0, 3) iterator`

لذا سوف نقوم بإعطائه ال **iterator** هكذا

```
nums.erase(nums.begin(), nums.begin() + 2);
cout << "First Element After Delete Is: " << *nums.begin() << "\n";
```

```
First Element After Delete Is: 30
```

**ملحوظة ٣:** آخر عنصر `but not including` لذا يقوم بحذف أول عناصر فقط



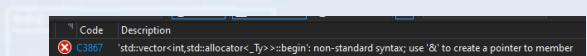
## #072 - Vector - Traversing With Iterator

```
Vector
- Iterator
--- Traversing
----- begin()
----- end()
----- advance()
```

- التنقل بين العناصر عن طريق ال **Iterator**

**ملحوظة:** يجب أن يكون النوع في ال **Iterator** مثل نوع ال **vector** وعدم وجود اختلاف حتى لا يحدث **error**

```
vector<int> nums = { 10, 20, 30, 40 };
vector<double>::iterator first = nums.begin();
```



**begin() - ١**

```
vector<int>::iterator first = nums.begin();
```

```
cout << "First Element Is: " << *first << "\n"; // 10
cout << "Second Element Is: " << first[1] << "\n"; // 20
cout << "Third Element Is: " << first[2] << "\n"; // 30
```

First Element Is: 10  
Second Element Is: 20  
Third Element Is: 30

**last() - ٢**

```
vector<int>::iterator last = nums.end() - 1;
```

```
cout << "Last Element Is: " << *last << "\n"; // 40
cout << "Before Last Element Is: " << *last - 1 << "\n"; // 39
cout << "Before Last Element Is: " << *(last - 1) << "\n"; // 30
```

Last Element Is: 40  
Before Last Element Is: 39  
Before Last Element Is: 30

**ملحوظة ٢:** يجب وضع **pranetheses** حتى يشير إلى العنصر الذي يسبقه في ال **vector** ولا يقوم بطرح واحد من الرقم الأخير بعد أن يشير إليه كما في المثال السابق

**advance - ٣:** يعني يتقدم وهو ان تقوم بالتقديم من عنصر لعنصر آخر داخل ال **vector** للأمام أو للخلف.

```
advance(first, 3);

cout << "First Element Is: " << *first << "\n"; // 40
```

First Element Is: 40

```
advance(first, -2);

cout << "First Element Is: " << *first << "\n"; // 20
```

First Element Is: 20



## #073 - Vector - Loop With Iterator And Ranged Loop

```
Vector
- Iterator
--- Loop With Iterator
--- Ranged Loop With For
```

### :Loop With Iterator - ١

```
vector<int> nums = { 10, 20, 30, 40 };
vector<int>::iterator it;

// Loop With Iterator
for (it = nums.begin(); it != nums.end(); ++it)
{
    cout << *it << "\n";
}
```

```
10
20
30
40
```

ملحوظة: من الممكن جعلها أصغر من `nums.end()` أو `for (it = nums.begin(); it < nums.end(); it++)`

`for (it = nums.begin(); it != nums.end(); it++)` Not Equal

ملحوظة ٢: من الأفضل ان نقوم بعمل `pre increment` لأنها أسرع لأنها لا تقوم بعمل

`for (it = nums.begin(); it != nums.end(); ++it)` وترجعه مباشرة `object copy` ومن الممكن أن نقوم بعمل `post increment` ولكنه أبطأ

```
for (it = nums.begin(); it != nums.end(); it++)
```

### :Ranged Loop With For - ٢

```
// Ranged Loop With For
for (int val : nums)
{
    cout << val << "\n";
}

cout << "===== \n";

int numbers[5] = { 20, 40, 60, 80, 100 };

for (int myNumber : numbers)
{
    cout << myNumber << "\n";
}
```

```
10
20
30
40
=====
20
40
60
80
100
```

## #074 - Vector - Use Iterator To Count, Sort & Reverse

```
Vector
- Use Iterator To:
--- Sort
--- Count
--- Reverse
```

ملحوظة: يجب أولاً أن نقوم باستدعاء ال `Header File` الخاص بال `algorithm` لنسخدمهم



## هي function Count - ١ تقوم بحساب عدد مرات تكرار رقم في ال vector

```
vector<int> nums = { 10, 500, 60, -20, 20, 20, 100, 20 };  
  
int val = 20;  
int countTimes = count(nums.begin(), nums.end(), val);  
  
cout << "Number " << val << " Found " << countTimes << " Times.\n";
```

```
Number 20 Found 3 Times.
```

**ملاحظة ٢:** من المفضل أن نقوم بوضع علامة [%] هنا لأنها تقوم بعمل access على القيمة مباشرة لذا فإنها أسرع

```
for (int &n : nums)  
{  
    cout << n << "\n";  
}
```

```
10  
500  
60  
-20  
20  
20  
100  
20
```

هي function sort - ٢ تقوم بترتيب عناصر ال vector من القيمة الأصغر للقيمة الأكبر

```
sort(nums.begin(), nums.end());  
  
for (int &n : nums)  
{  
    cout << n << "\n";  
}
```

```
-20  
10  
20  
20  
20  
60  
100  
500
```

هي function reverse - ٣ تقوم بعكس ترتيب عناصر ال vector

```
vector<int> nums = { 10, 500, 60, -20, 20, 20, 100, 20 };  
  
reverse(nums.begin(), nums.end());  
  
for (int &n : nums)  
{  
    cout << n << "\n";  
}
```

```
20  
100  
20  
20  
-20  
60  
500  
10
```



## #075 - Pointers - What Are Pointers?

```

Pointers

What Are Pointers?
--- A Variable That Store Memory Address Of Other Variable

Why We Need Pointers?
--- To Iterate On Elements In Data Structures Like Array
--- Pass Function To Other Function
--- Dynamic Memory Allocation

Benefits Of Using Pointers
--- Reduce The Code and Increase Performance

Note
--- There's Raw Pointer And Smart Pointer

Syntax
--- Declare A Pointer
--- Print Variable Memory Address => Reference Operator || Address of [&]
--- Print Value That Memory Address Point To => Dereference Operator [*]
--- Change Variable Value With Pointer

```

**Memory Address**: معناها مؤشر وهو عبارة عن **Variable** يخزن قيمة ال **Object** أو **Variable** الخاص به.

**ملحوظة:** نقوم بوضع علامة ال **[&]** قبل اسم المتغير التي تسمى **Reference** and **Address Of Operator**

مثل هذا المثال: نقوم بأخبار ال **compiler** بأن هذا ال **pointer** يساوي **Memory**

**int* ptr = &num;** **Address of "num"**

```

30 int num = 100;
31 int* ptr = &num;
32
33 cout << "Value: " << num << "\n";
34 cout << "Address: " << &num << "\n";
35 cout << "Address: " << ptr << "\n";

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Address: 0x79513ffb84  
Address: 0x79513ffb84

**ملحوظة ٢:** ولكي نصل إلى القيمة الموجودة في هذا ال **Address** نقوم باستخدام ال **Deference Operator [ * ]**

```

30 int num = 100;
31 int* ptr = &num;
32
33 cout << "Value: " << num << "\n";
34 cout << "Address: " << &num << "\n";
35 cout << "Address: " << ptr << "\n";
36 cout << "Value: " << *ptr << "\n";

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Value: 100  
Address: 0x40bd9ff984  
Address: 0x40bd9ff984  
Value: 100



## - لكي نصل للعنصر ونغير القيمة الموجودة داخل الـ **Memory Address** نقوم بالتعديل علي الـ **Deference Operator Of Memory Address**

**ولن يتغير الـ **Memory Address** لكن تتغير القيمة**

```

30 int num = 100;
31 int* ptr = &num;
32
33 cout << "Value: " << num << "\n";
34 cout << "Address: " << &num << "\n";
35 cout << "Address: " << ptr << "\n";
36 cout << "Value: " << *ptr << "\n";
37
38 *ptr = 200;
39
40 cout << "Value: " << num << "\n";
41 cout << "Address: " << &num << "\n";
42 cout << "Address: " << ptr << "\n";
43 cout << "Value: " << *ptr << "\n";

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```

Value: 100
Address: 0xd9aa7ff784
Address: 0xd9aa7ff784
Value: 100
Value: 200
Address: 0xd9aa7ff784
Address: 0xd9aa7ff784
Value: 200

```

## #076 - Pointers - Pointing To Array

**pointer عن طريق الـ **Array** : أن نشير إلى عنصر في الـ **Array** عن طريق الـ **pointer****

```

int nums[]{10, 20, 30, 40};
int *ptr = &nums[0];

cout << "First Element\n\n";
cout << "Value With Index: " << nums[0] << "\n";
cout << "Value With Pointer: " << *ptr << "\n";
cout << "Address With Index: " << &nums[0] << "\n";
cout << "Address With Pointer: " << ptr << "\n";
cout << "=====\n";
cout << "Second Element\n\n";
cout << "Value With Index: " << nums[1] << "\n";
cout << "Value With Pointer: " << *(ptr + 1) << "\n";
cout << "Address With Index: " << &nums[1] << "\n";
cout << "Address With Pointer: " << ptr + 1 << "\n";
cout << "=====\n";
cout << "Third Element\n\n";
cout << "Value With Index: " << nums[2] << "\n";
cout << "Value With Pointer: " << *(ptr + 2) << "\n";
cout << "Address With Index: " << &nums[2] << "\n";
cout << "Address With Pointer: " << ptr + 2 << "\n";

```

++\#076 - Pointers - Pointing To Array  
Elzero Web School\#2 - Fundamentals  
Array\app  
First Element  
  
Value With Index: 10  
Value With Pointer: 10  
Address With Index: 0x1705bfffbc0  
Address With Pointer: 0x1705bfffbc0  
=====  
Second Element  
  
Value With Index: 20  
Value With Pointer: 20  
Address With Index: 0x1705bfffbc4  
Address With Pointer: 0x1705bfffbc4  
=====  
Third Element  
  
Value With Index: 30  
Value With Pointer: 30  
Address With Index: 0x1705bfffbc8  
Address With Pointer: 0x1705bfffbc8  
  
[Done] exited with code=0 in 0.482 s  
[Running] cd "f:\Programming\4- Elze  
rzo\#076 - Pointers - Pointing To Ar

**ملحوظة:** في العنصر الثاني والثالث نقوم بوضع الـ **pointer** داخل **prantheses** لأننا عندما نقوم بعمل **(ptr + 1)** نخبره بأن يذهب للـ **Memory Address** الذي يليه ثم نقوم باستخدام الـ **[*]** **deference operator** لكي يحضر القيمة **Memory Address** في هذا الـ **Memory Address**

**ملحوظة ٢:** نوع البيانات **integer** له **size of integer** يكون **4 Bytes**



## ملاحظة ٣: عندما نقوم بوضع short تتحول من 2 Bytes إلى 4 Bytes

```
short int nums[4]{10, 20, 30, 40};
short int *ptr = &nums[0];

cout << "First Element\n\n";
cout << "Value With Index: " << nums[0] << "\n";
cout << "Value With Pointer: " << *ptr << "\n";
cout << "Address With Index: " << &nums[0] << "\n";
cout << "Address With Pointer: " << ptr << "\n";
cout << "=====\n";
cout << "Second Element\n\n";
cout << "Value With Index: " << nums[1] << "\n";
cout << "Value With Pointer: " << *(ptr + 1) << "\n";
cout << "Address With Index: " << &nums[1] << "\n";
cout << "Address With Pointer: " << ptr + 1 << "\n";
cout << "=====\n";
cout << "Third Element\n\n";
cout << "Value With Index: " << nums[2] << "\n";
cout << "Value With Pointer: " << *(ptr + 2) << "\n";
cout << "Address With Index: " << &nums[2] << "\n";
cout << "Address With Pointer: " << ptr + 2 << "\n";
cout << "=====\n";
cout << "Fourth Element\n\n";
```

Elzero Web School\#2 -- Fundamentals  
Array\app  
First Element  
  
Value With Index: 10  
Value With Pointer: 10  
Address With Index: 0x2b131ff780  
Address With Pointer: 0x2b131ff780  
=====  
Second Element  
  
Value With Index: 20  
Value With Pointer: 20  
Address With Index: 0x2b131ff782  
Address With Pointer: 0x2b131ff782  
=====  
Third Element  
  
Value With Index: 30  
Value With Pointer: 30  
Address With Index: 0x2b131ff784  
Address With Pointer: 0x2b131ff784  
=====  
Fourth Element  
  
Value With Index: 40  
Value With Pointer: 40  
Address With Index: 0x2b131ff786  
Address With Pointer: 0x2b131ff786  
  
[Done] exited with code=0 in 0.466 s

## #077 - Pointers - Void And Wild Pointer And Null

Wild Pointer - ١: هو ال الذي لا يشير لـ Memory Address خاصة بمتغير آخر

ملاحظة: عندما نحاول طباعة الـ Wild Pointer ينتج Garbage Value مثل عندما ننشئ متغير ولا نعطيه قيمة

```
22 int *ptr1; // Wild
23
24 cout << ptr1 << "\n"; // Garbage Value
```

0x2860f9816b0

ملاحظة ٢: وعندما لا نريد أن نعطيها قيمة Garbage Value نقوم بوضع NULL أو nullptr في القيمة ومن ثم نعطيها القيمة 0

```
18 int *ptr1; // Wild
19 int *ptr2 = NULL;
20 int *ptr3 = nullptr;
21
22 cout << ptr1 << "\n"; // Garbage Value
23 cout << ptr2 << "\n"; // 0
24 cout << ptr3 << "\n"; // 0
```

0x83  
0  
0



**ملحوظة ٣:** عندما نريد أن نوجه الـ **pointer** إلى بيانات لا نعرف نوعها سوف نقوم باستخدام الـ **Void Array**

ولكن لا نستطيع أن نصل للقيمة عن طريق الـ **Deference Operator** سوف يعطي **error** إذا حاولنا بذلك بسبب اختلاف أنواع البيانات

```
int a = 100;
void *ptr = &a;
cout << *ptr << "\n";
```

'void*' is not a pointer-to-object type gcc  
View Problem (Alt+F8) No quick fixes available

وسوف نقوم بحل هذه المشكلة عن طريق الـ **Cast** لـ **pointer** لنوع البيانات المناسب :**Casting**

**C-Style - ١**

```
25 // C-Style
26 cout << *(int *)ptr << "\n"; // 100
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

100

**Modern - ٢**

```
28 // Modern
29 cout << *static_cast<int *>(ptr) << "\n"; // 100
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

100

## #078 - Pointers - Arithmetic And Array

Pointers  
-- Pointer Arithmetic  
-- Pointer And Array

**ملحوظة:** عندما نقوم بإنشاء **Array** ومحاولة طباعتها سوف يعطينا الـ **Memory Address** لأول **element** في الـ **Array**

```
22 int nums[]{10, 20, 30, 40, 50};
23 cout << nums << "\n"; // 1st Element ==> Memory Address
24 cout << &nums[0] << "\n"; // 1st Element ==> Memory Address
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

0x78501ff8b0  
0x78501ff8b0



## Pointer Arithmetic - Bashtakad al square brackets

```
20 cout << nums[0] << "\n"; // 1st Element => 10
21 cout << *nums << "\n"; // 1st Element => 10
22
23 cout << nums[1] << "\n"; // 2nd Element => 20
24 cout << *(nums + 1) << "\n"; // 2nd Element => 20
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
10
10
20
20
```

## Arithmetic Operator With Pointer -

```
22 int *ptr = nums;
23
24 cout << ptr << "\n"; // 1st Element => Memory Address
25 cout << *ptr << "\n"; // 1st Element => 10
26
27 ptr++;
28
29 cout << ptr << "\n"; // 2nd Element => Memory Address
30 cout << *ptr << "\n"; // 2nd Element => 20
31
32 ptr += 3;
33
34 cout << ptr << "\n"; // Last Element => Memory Address
35 cout << *ptr << "\n"; // Last Element => 50
36
37 ptr--;
38
39 cout << ptr << "\n"; // Before Last Element => Memory Address
40 cout << *ptr << "\n"; // Before Last Element => 40
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
0xf6315ffe20
10
0xf6315ffe24
20
0xf6315ffe30
50
0xf6315ffe2c
40
```