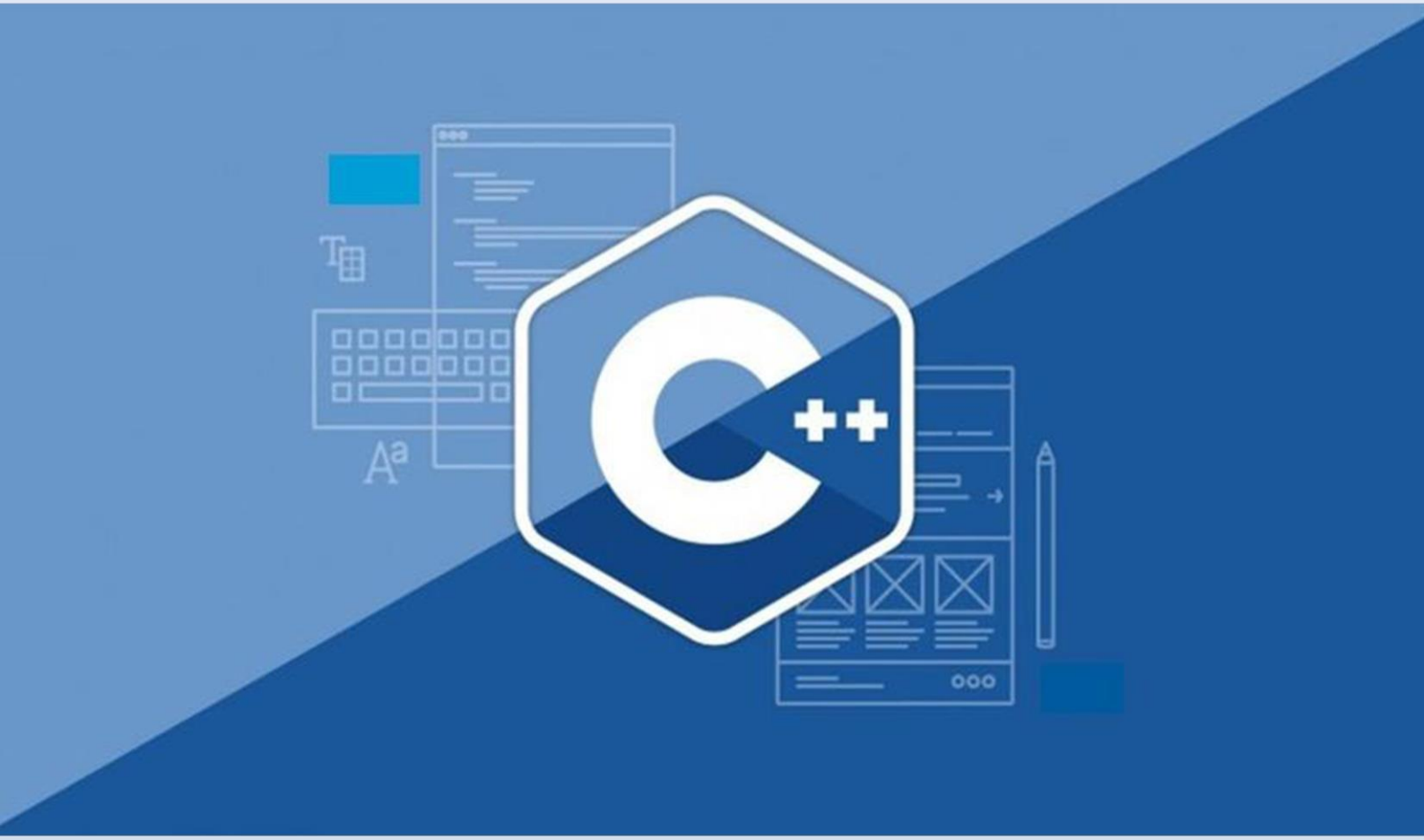
# Fundamentals Of Programming with C++

## ملخص كورس Fundamentals Of Programming With C++



A lot of thanks to Osama Elzero

Created By: Fady Alamir

Facebook: fadyalamir77 | LinkedIn: fady-alamir77 | GitHub: fadyalamir

# #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 مفهمتش بردك اسأل وطبيعي إنك متفهمش من اول مرة

٤- اسمع كلام مدرب الكورس لو قلك اتفرج على فديو 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 علي اكثر 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 لنستخدم الـ Functions الموجودة بها في الملف.

٢- الـ Source file: هو الملف الذي نقوم داخله بكتابة الكود.

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

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

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

٦- iostream: هو عبارة عن file بداخله مجموعة من الـ Functions نحتاج الي 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 الـ flow, stream ستقرر من الذي يـ Run الأول.

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

## #006 - Preprocessing, Compiling And Linking

١- الملف iostream نقوم باستدعائه ويسمي header file امتداده يكون h. أو امتدادات أخري 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 لـ compiler حيث أن الكود او المحتوي هو الأهم.

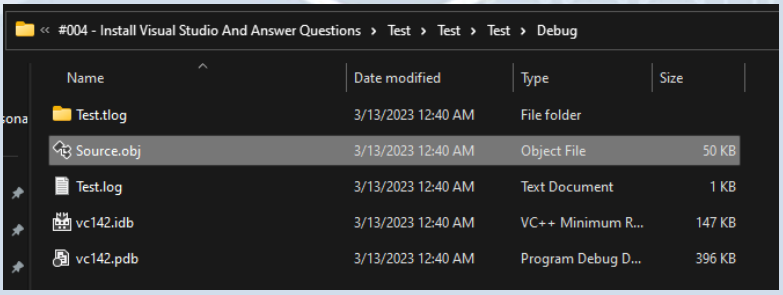
Fady.jpg

F: > Programming > 4- Elzero Web School > #2 - Fundamentals Of Pro

```cpp
#include <iostream>

int main()
{
    std::cout << "Line 1\n";
    std::cout << "Line 2\n";
    std::cout << "Line 3\n";
    return 0;
}
```

٣- انشاء ملف الـ Translation Unite سوف يكون بداخله محتوي 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.




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 نقوم 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 الـ { }: هما الـ containers أي الحاوية التي تحتوي علي Block of code صندوق الأكواد.

٣- كل سطر برمجي ينتهي بـ ; أي semicolon

٤- هذه العلامة : تسمي colon وعندما تكون :: تسمي double colon

٥- cout: اختصار لـ character output

٦- Output: المخرجات ٧- Input: المدخلات

٨- كل حرف او علامة يُسمي character في الـ code والـ space المسافة الفارغة تسمي character

٩- << ← stream insertion operator حيث أن هذا الـ operator مسؤول عن الادخال يرسل البيانات التي تليه للـ function لتخرج للـ stream

١٠- Syntax highlighter: يقوم بتميز كل كود بلونه المميز.

١١- "Line\n" ← نوع من البيانات يسمي string ويضاف داخل الـ double quotes

١٢- أي شيء يأتي بعد الـ \ أي backslash end يسمي special character أي يقوم بعمل شيء مميز مثل \n مسؤول على أن يقوم بعمل new line

١٣- 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 comment بعد double forward slash - //

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

٢- Multiple line comment

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

ويمكن 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 Ctrl + / يمكن عمل comment or uncomment للسطر أو لعدة أسطر بعد تحديدهم

## #009 - Variables Basic Knowledge

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

٢- أي شيء داخل الـ Double Quotes أي rapped in double quotes سوف تطبع كما هي بدون تطبيق العمليات الحسابية أو أي شيء اخر 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- Declare a variable: انشاء متغير أو التصريح عن وجود متغير جديد أو الإعلان عن وجود متغير جديد.

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

٤- نوع المتغير int: أي Integer وهو رقم 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 الأخرين 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
--- Reserverd 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 أخر Price لا يحدث error 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 underscore في المنتصف أو في أخر اسم الـ variable بدون وجود errors.

والـ underscore يمكن وضعها في بداية اسم الـ variable بدون error ايضاً.

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

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
100
```

٥. No White Space Or Special Characters: عند استعمال white space في اسم أي Identifier يعتبروا 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 - Va
app.cpp: In function 'int main()':
app.cpp:30:15: error: expected initializer before 'bers_'
   30 |     int _num_ bers_  = 100;
      |               ^~~~
```

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 Wit
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;
      |                    ^
```

٦. Reserved Keywords "Class, Public": يوجد كلمات محجوزة في اللغة لا يمكن استخدامها في تسمية الـ variable

**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;
      |         ^~~~~~
```

٧. Writing Styles:

مقدمة

هذه بعض أشهر طرق كتابة ال Code مع بعض اللغات التي تستخدم هذه الطريقة

| 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# |

٨. Related Names: أسم معبر عن المتغير يكون مفهوم.

## #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
```

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

```
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:\Programmin
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
```

٢- Local Variable: هي التي توجد داخل function واحدة معينة ولا يستطيع أي function أخر أن يـ access عليه

```cpp
int a = 100; // Global Variable

int second()
{
    cout << a << " Coming From Second Function\n";
    cout << b << " Coming From Second Function\n";
    return 0;
}

int main()
{
    int b = 200; // Local Variable
    cout << a << " Coming From Main Function\n";
    cout << b << " Coming From Main Function\n";
    second();
    return 0;
}
```

```
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";
      |             ^
```

```cpp
int a = 100; // Global Variable

int second()
{
    int b = 200; // Local Variable
    cout << a << " Coming From Second Function\n";
    cout << b << " Coming From Second Function\n";
    return 0;
}

int main()
{
    cout << a << " Coming From Main Function\n";
    cout << b << " Coming From Main Function\n";
    second();
    return 0;
}
```

```
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
```

١- const int: هو متغير ثابت للقراءة فقط لا يمكن تغيره او التعديل عليه.

```cpp
int main()
{
    const int num = 100;
    num = 200;
    cout << num;
    return 0;
}
```

→

```
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 اخري. const int x

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

٣- يوجد طريقة أخري لأنشاء متغير ثابت منها بواسطة preprocessor directive

```cpp
#define DAYS 9
```

→

```cpp
    cout << "\n"
         << DAYS;
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TER
[Running] cd "f:\Programm
Variables - Constant Vari
Fundamentals Of Programmi
100
9
```

## #014 – Calculate Your Age Application

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

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

## #015 - Escape Sequences Characters

```
Escape Sequances 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
```

Escape Sequences Characters: يوجد بعض الحروف المميزة التي من خلالها نستطيع أن نتحكم في السلوك الخاص بالطباعة.

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

٢- Control Printing Behavior: نتحكم في سلوك الطباعة

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

```
main()
cout <<
cout <<
cout << "Line 3\"
```

missing terminating " character gcc

missing terminating " character gcc

View Problem (Alt+F8) No quick fixes available

٤- Can Be Interested In Any Position: يمكن استعمالها في أي مكان في الكود.

٥- لطباعة \ في الـ stream نستخدم \\ double back slash هكذا حيث يقوم بعمل skip للذي بعده.

```
int main()
{
    cout << "Line 1\n";
    cout << "Line 2\n";
    cout << "Line 3\\";
    return 0;
}
```

```
[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 double quotes في الـ stream لا يمكن اضافتها هكذا

```
    cout << "Line "4"";
    return 0;
}
```

```
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 الـ compiler تعامل مع الـ double quotes برمجياً وليس كأنها مكتوبة للطباعة بحيث أن Line قبلها start quote وبعدها closing quote والـ 4 خارجهم لذا يحدث error ويحتاج إلي ; semi colon بعد الـ 4 ويمكن حل هذه المشكلة هكذا بوضع back slash قبل كل من الـ start quote والـ closing quote.

```
int main()
{
    cout << "Line 1\n";
    cout << "Line 2\n";
    cout << "Line 3\\\n";
    cout << "Line \"4\"\n";
    return 0;
}
```

```
Line 1
Line 2
Line 3\
Line "4"
```

٧- تستخدم الـ back slash للـ single quote أيضاً.

٨- الـ tab وهي عبارة عن 8 white space وتطبع عن طريق \t وتسمي special skipping character ويمكن تغير الـ spaces من الـ text editor أو الـ IDE

```
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\t\t7\n";
    return 0;
}
```

```
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 الـ back space أي يقوم بمسح الذي قبله إذا كان حرف او white space

```
30      cout << "Line\b8"; // Lin8
31      return 0;
32  }
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
Lin8
```

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

١١- \r الـ Carriage Return

```
32      cout << "Fady\rAlamir"; // Alamir
33      return 0;
34  }
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
Alamir
```

مثل Fady\rAlamir تقوم أولاً بطباعة Fady ثم الـ Carriage Ruturn تقوم بإرجاع الـ mouse cursor عند حرف الـ F ثم يقوم بعمل override علي Fady ويطبع ما بعد الـ \r مكانها Fady ⟵ Alamir

## #016 - Data Types - What Is Data?

```
    Data Types
    - What Is Data?
    - Data Examples In Real Life
    --- Integer => 5000, 10, -100
    --- Srting => "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، Data

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

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 المُدرس واسم الفصل

٣. Boolean: تعبر عن أي إجابة بها Yes, No أو أي switch به On, Off أو ١ ، ٠ أو True, False

٤. Float (floating point number) OR Double

الأرقام التي بها كسور

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
```

٢. Operation: كل نوع من أنواع البيانات نختاره 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:
'std::__cxx11::basic_string<char>'})
   35 |     cout << num_one / name; // Error
      |             ~~~~~~~ ^ ~~~~
      |             |       |
      |             int     std::string {aka std::__cxx11::basic_string<char>}
```

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

## #017 - Data Types, Sizes And Memory

١- Ram ([R]andom [A]ccess [M]emory): ذاكرة الكمبيوتر بها مجموعة من الأماكن نخزن بها البيانات.

٢- Data Sizes – أحجام البيانات:

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.

٣. Tera Byte => 1024 Gigabytes

٤. Giga Byte => 1024 Megabytes

٥. Mega Byte => 1024 Kilobytes

٦. Kilo Byte => 1024 Byte

٣- Data Types With Size – أنواع البيانات وأحجامها:

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

٢. float => 4 Bytes [18.5656565656]

٣. double => 8 Bytes [18.5656565656]

- Number . Fractional Number => أي الجزء الذي بعد الكسر يُسمي مكون الكسر.

- الفرق بين Float & Double

١. Float: يمكن استخدام ٧ أرقام كـ كسور

٢. Double: يمكن استخدام ١٥ رقم كـ كسور أي Double the float

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 حجز عدد معين من الـ Bytes في الـ Memory بناءً على نوع البيانات ثم يقوم بوضع القيمة التي سوف تقوم بوضعها في المكان الذي قمت بتحديده في الـ Memory.

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

٥- يمكن أن تجعل الـ compiler يقوم باستنتاج نوع المتغير من القيمة التي اعطتها للمتغير عن طريق وضع كلمة auto قبل أسم الـ 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 الـ compiler.

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

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
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 أسم المتغير بعد cout وجعل الـ compiler يقوم بطباعة موقع المتغير في الذاكرة.

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

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TE
0x78d47ffd68
```

## #018 - Data Types – Integer

١- Primitive Data Types – أنواع البيانات الأولية

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

٢. عند وضع true أو false لـ Int يعطي ١ لقيمة الـ 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 بدلاً من الاعداد الصحيحة.

```
25    int num_six = "Fady";
26
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
app.cpp: In function 'int main()':
app.cpp:25:17: error: invalid conversion from 'const char*' to 'int' [-fpermissive]
   25 |   int num_six = "Fady";
      |                 ^~~~~~
      |                 |
      |                 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 header file يسمي limits.h يوجد بداخله الـ INT_MIN والـ INT_MAX

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    return 0;
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 = تسمي Assignment Operator أي تقوم بـ assign الـ value للمتغير num_one أي تعيين 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 Performance وحجم الـ Bytes المحددة في الـ Memory.

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   return 0;
25 }
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
20
8
20.5
```

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

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



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

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

٤- وعند وضع الـ f بعد الرقم مع نوع المتغير auto يجعلها float ايضاً

```
27   cout << fl << endl;
                (float)(10.5F)
28
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- Character Data Type: الـ character نقوم بتخزين حرف واحد فيه فقط ويحجز 1 Byte من الرام والقيمة توضع داخل single quotes

٢- Type Casting: نقوم بوضع int (‘%’) وينتج لنا الـ ASCII Number لعلامة %

```
31   cout << int('%') << "\n"; // 37
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
37
```

وعندما نقوم بوضع double quotes يحدث error لأن array of characters to int لا ينفع.

٣- يمكن انتاج الحرف باللغة الإنجليزي عن طريق رقمه في الـ ASCII Table هكذا

```
37    cout << char(81) << "\n"; // Q
38
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
Q
```

## #021 - Data Types - Bool And Void

### ١- Boolean:

١. قيم الـ Boolean عبارة عن ture, false جميع الحروف small لأن الحروف sensitive أو أي إجابة بها Yes, No أو أي switch به On, Off أو ١ ، ٠

٢. نستخدم true, false ولا نقوم باستخدام 1, 0 لأن الـ Boolean للـ true false , تقوم بحجز 1 Byte فقط لكن الـ 1, 0 عبارة عن integer تقوم بحجز 4 Byte لذا سوف يكون الـ Performance أقل إذا استخدمنا الـ 1, 0.

```
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 الـ Boolean الـ true, false فقط ولكن من الممكن أن نقوم باستخدام معادلات او معادلة او 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 لل Boolean يكون الناتج True أي ١ وعندما لا نضع قيمة يكون الناتج False أي ٠

```
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 الـ function.

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

## #022 - Data Types - Modifiers And Type Alias

١- نقوم باستخدام short int عندما لا نحتاج لل max int مثل العمر حيث أن العمر لن يصل إلي هذا الرقم `#define SHRT_MAX 32767` وتقليل حجم الـ Bytes المحجوزة في الرام من 4 Bytes لـ 2 Bytes مما يزيد من performance البرنامج.

الـ short int هو integer أقصر من الـ integer الطبيعي.

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

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
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
2
```

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

٣- long long: يمكنك من استخدام رقم بهذا الحجم `LLONG_MAX 9223372036854775807i64`
لذا يمكننا التحكم في الـ modifier

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

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

٦- Unsigned [int, char]: يُخزن بها أرقام موجبة فقط

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

٧- 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- Type Conversion – تحويل أنواع البيانات:

أي تحويل البيانات من نوع لنوع أخر، ويوجد نوعين من التحويل

### ١. Implicit Conversion – التحويل الضمني:

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
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Code

```
20
4
```

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

```
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. Explicit Conversion – التحويل الصريح:

التحويل يتم عن طريق الـ programmer وتسمي هذه العملية احياناً بالـ 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 لـ Type Casting هنا في المثال السابق تسمي function like

**ملحوظة:** عند تحويل نوع من أنواع البيانات يحدث فقدان للبيانات، وهذا منطقي وطبيعي لأن عندما نقوم بتحويل مثلاً من **long long** إلى **short int** من الطبيعي ان 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
```

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

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

٣- **Arithmetic Operators**: الـ **operators** الخاصة بالعمليات الحسابية.

١. **Addition**:

```
23    cout << 10 + 10 << "\n";
24    cout << 10.5 + 9.5 << "\n";
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
20
20
```

٢. **Subtraction**:

```
33    cout << 100 - 50 << "\n";
34    cout << 100 - -50 << "\n";
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
50
150
```

٣. Multiplication:

```
36    cout << 10 * 5 << "\n";
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
50
```

٤. Division:

```
38    cout << 20 / 5 << "\n";
39    cout << 12 / 5 << "\n";
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
4
2
```

```
38    cout << 20 / 5 << "\n";
39    cout << 12.f / 5.f << "\n";
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
4
2.4
```

**ملحوظة:** يجب وضع f. لترمز عن الـ float لكي ينتج القيمة بالكسر لان ناتج قسمة int/int ينتج int أي عدد صحيح.

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

```
38    cout << 20 / 5 << "\n";
39    cout << 12 / 5 << "\n";
40    cout << 12.f / 5.f << "\n";
41    cout << 12.f / 5 << "\n";
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
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    cout << 20 % 5 << "\n";
48    cout << 21 % 5 << "\n"; // 1
49    cout << 24 % 5 << "\n"; // 1
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
0
1
4
```

ملحوظة: الـ Modulus لا يتعامل إلا مع integer

```
50    cout << 24.5 % 5 << "\n";
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Code

```
app.cpp: In function 'int main()':
app.cpp:50:16: error: invalid operands of types 'double' and 'int' to binary 'operator%'
   50 |    cout << 24.5 % 5 << "\n";
      |            ~~~~ ^ ~
      |            |     |
      |            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 للـ value التي على اليسار للـ variable التي على اليمين.**

١- **Assign: هي علامة الـ = التي نستخدمها لوضع قيمة في variable هكذا** `int a = 10;`

٢- **Addition: معناها ان الـ compiler يقوم باستخدام الـ variable ويجمع القيمة التي بعد الـ =+ على قيمة الـ variable هكذا**

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
25
```

٣- **Subtraction: معناها ان الـ compiler يقوم باستخدام الـ variable ويطرح القيمة التي بعد الـ =- من قيمة الـ 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
```

٤- Multiplication: معناها ان الـ compiler يقوم باستخدام الـ variable 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
```

٥- Division: 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
```

٦- Modulus: 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

١- Pre: 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 يسمي Pre Increment بحيث يزود الـ 1 وبعدها يقوم بطباعة الـ 1

٢- Post: شيء يحدث بعد عملية الـ execute

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
0
1
```

هذا يسمي Post Increment بحيث يطبع الـ 0 وبعدها يقوم بزيادة 1 عليها

٣- Post Decrement: يقوم بطباعة الـ 0 وبعدها يقوم بطرح 1

٤- Pre Decrement: يقوم بطرح 1 وبعدها يقوم بطباعة الـ 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
```

- Comparison Operators: الـ Operators المسؤولة عن المقارنة لمقارنة القيم ببعض واحياناً تسمي الـ Relational Operators أي الـ Operators التي تجعلك تري علاقة القيم ببعضها البعض.

١- Equal (==): تستخدم لمقارنة قيمتين ببعض.

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
1
0
```

٢- Not Equal(=!): لمقارنة القيمتين وسؤال الـ compiler هل هذه القيمة لا تساوي القيمة الأخرى.

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
0
1
```

٣- Greater Than(>): للسؤال هل قيمة أكبر من قيمة أخري

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
1
0
```

٤- Less Than(>): للسؤال هل قيمة أصغر من قيمة أخري

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
1
```

٥- Greater Than Or Equal(=>): للسؤال هل قيمة أكبر من او يساوي قيمة أخري وعند تحقق شرط من الشرطين ينتج True

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
1
1
```

٦- Less Than Or Equal(=>): للسؤال هل قيمة أصغر من أو يساوي قيمة أخري وعند تحقق شرط من الشرطين ينتج True

```
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
```

**Logical Operators: هي الـ Operators التي من خلالها نستطيع أن نفحص المنطق بين القيم الخاصة بنا "نطلب أكثر من طلب".**

**١. And(&&): هذا الـ Operator يتكون من علامتين Ampersand أو تسمي "و" العطف ومعناها and نستخدمها لطلب طلبين 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  DEBUG CONSOLE  TERMINAL
1
```

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

**٢. Or(||): هذا الـ Operator 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;
27    int points = 800;
28    cout << (age >= 18 || points >= 500) << endl; // 1 => True
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
1
```

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

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

٣. Not(!): 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- Operators Precedence: أي أن Operator له الأولية أن يعمل قبل Operator أخر 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: Operator 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
```

مثال٢: 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
```

مثال٣: 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 ان تتم العملية من الشمال لليمين بدون ترتيب الأوليات لل Operators يجب أن تقوم بعزل عملية الجمع أو الطرح بوضعها في أقواس "parentheses" هكذا

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
75
```

## #030 - Control Flow - If Condition Introduction

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

١. if: الـ Syntax 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

**١- If: هو الشرط الأول عند تحققه يُنفذ ولا ينظر لباقي الشروط else if, else**

```cpp
int main()
{
  int age = 25;
  int points = 450;
  int rank = 4;

  if (age >= 18)
  {
    cout << "Welcome Your Age Is Ok\n";
  }
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
Welcome
```

**٢- else if: يتحقق ويُنفذ عندما لم يتحقق الشرط الأول if**

```cpp
int age = 15;
int points = 800;
int rank = 4;

if (age >= 18)
{
  cout << "Welcome Your Age Is Ok\n";
}
else if (points > 500)
{
  cout << "Welcome Your Points Is Ok\n";
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
Welcome Your Points Is Ok
```

```cpp
int age = 15;
int points = 450;
int rank = 8;

if (age >= 18)
{
  cout << "Welcome Your Age Is Ok\n";
}
else if (points > 500)
{
  cout << "Welcome Your Points Is Ok\n";
}
else if (rank > 5)
{
  cout << "Welcome Your Rank Is Ok\n";
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
Welcome Your Rank Is Ok
```

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

```cpp
int age = 15;
int points = 450;
int rank = 4;

if (age >= 18)
{
  cout << "Welcome Your Age Is Ok\n";
}
else if (points > 500)
{
  cout << "Welcome Your Points Is Ok\n";
}
else if (rank > 5)
{
  cout << "Welcome Your Rank Is Ok\n";
}
else
{
  cout << "Iam Sorry\n";
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
Iam Sorry
```

## #032 - Control Flow - Nested If Conditions

**Nested If Conditions: أي حالات شرط Conditions متداخلة أي condition داخل condition أخري، تستخدم لعمل filtration لمجموعة من البيانات أي تـ filter من 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

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

```
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;
```

١- Nested Ternary Operator: كتابة الشرط وفي حالة الـ True يطبع الناتج المكتوب في الـ Code

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

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

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

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

٢- عندما يكون الـ Block of Code في الـ Conditions عبارة عن one line 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 بإضافة ٣ متغيرات بجانب بعض بهذه الطريقة `cin >> a >> b >> c;` `cout << a << b << c;`

**Done Applications**

## #036 - Control Flow - Switch Case

- Switch: معناها التحويل أو التبديل بين شيء وشئ أخر

- الـ Syntax:

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

default:
  break;
}
```

- الفرق بين if ، switch

```
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 لل switch وتنهي الـ cycle في هذا الجزء وإذا لم يكن الـ case هو 1 يبدأ يدخل علي case 2 وإذا لم يكن 2 أو 3 يرجع لل default

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

```
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";
}
```

```
PROBLEMS 2  OUTPUT  TERMINAL  DEBUG CONSOLE
PS C:\Users\ss> & 'c:\Users\ss\.vscode\extensions\ms-vscode.cpptools-1.15.4-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-id-2ibdpbw2.1if' '--stdout=Microsoft-MIEngine-Out-dbgExe=C:\msys64\mingw64\bin\gdb.exe' '--interpreter=mi'
Choose A Day From 1 To 25
1
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";
```

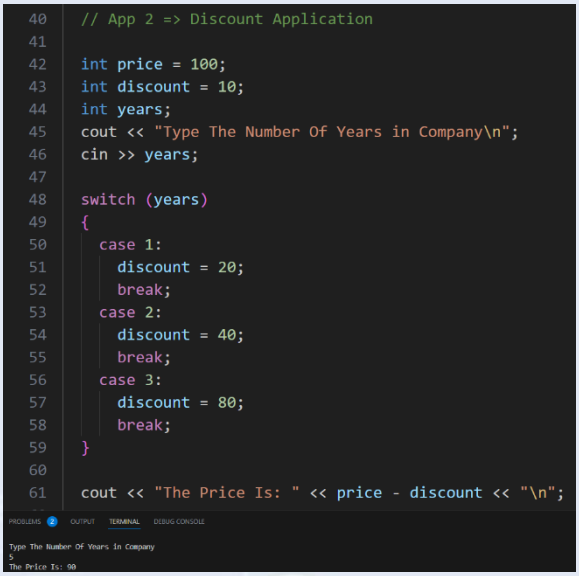
- الـ switch لا يقبل نوع بيانات غير integer, character

```
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
View Problem (Alt+F8)  No quick fixes available
{
  case 10.5:
```

## #037 - Switch Training - Create Three Application

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



```cpp
// App 2 => Discount Application

int price = 100;
int discount = 10;
int years;
cout << "Type The Number Of Years in Company\n";
cin >> years;

switch (years)
{
  case 1:
    discount = 20;
    break;
  case 2:
    discount = 40;
    break;
  case 3:
    discount = 80;
    break;
}

cout << "The Price Is: " << price - discount << "\n";
```

```
PROBLEMS 2  OUTPUT  TERMINAL  DEBUG CONSOLE

Type The Number Of Years in Company
3
The Price Is: 90
```

## #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:**

**١- 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 "0 based indexing" 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 علي جميعهم مع بعض بدلاً من وضع كل اسم أو رقم في متغير معين.

- Syntax of Array

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

- حجم المصفوفة Array Size:

يكون علي حسب نوع البيانات ويكون حاصل ضرب عدد البيانات*حجم نوع البيانات بالـ 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 تلقائياً

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

ملحوظة٢: ومن الممكن أيضاً كتابة Syntax الـ Array 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 علي 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 علي 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
```

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

```cpp
int nums[4];

nums[3] = 400; // Last Element
nums[2] = 300; // Third Element
nums[0] = 100; // First Element
nums[1] = 200; // Second Element

cout << "Elemnt 1: " << nums[0] << "\n";
cout << "Elemnt 2: " << nums[1] << "\n";
cout << "Elemnt 3: " << nums[2] << "\n";
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

```cpp
int nums[4];

nums[3] = 400; // Last Element
nums[2] = 300; // Third Element
// nums[0] = 100; // First Element
nums[1] = 200; // Second Element

cout << "Elemnt 1: " << nums[0] << "\n";
```

```
PROBLEMS 2  OUTPUT  TERMINAL  DEBUG CONSOLE
Elemnt 1: 2099648152
```

٣- عند تحديث الـ Index للمرة الثانية عند استخراجه ينتج في الـ Terminal الرقم الجديد بعد التحديث

```cpp
nums[1] = 1000; // Second Element

cout << "Element 2: " << nums[1] << "\n";
```

```
PROBLEMS 2  OUTPUT  TERMINAL  DEBUG CONSOLE
Element 2: 1000
```

## ٢- Get Length Of Array With Sizeof

```cpp
int anums[] = {100, 200, 300, 400, 500, 600}; // 24 / 4 = 6
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
```

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

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

```cpp
int main()
{
  int points_a[3] = {1, 2, 3};
  int points_b[3] = {4, 5, 6};
  int points_c[3] = {7, 8, 9};

  // Good 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

  return 0;
}
```

```
PROBLEMS 2  OUTPUT  TERMINAL  DEBUG CONSOLE
6
7
9
```

```cpp
// 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 ان يكون الـ columns ، int rows ثابتين const حتي لا يحدث error لأن إذا لم يكونوا ثابتين فيمكن تحديث صفوف وأعمدة الـ Array ولا ينفع حدوث هذا

```cpp
// 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

١- طريقة أخري لأنشاء الـ Array

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

```cpp
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
```

٢- لمعرفة حجم الـ Array نقوم باستخدام .size()

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

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
Elements Count: 4
```

٣- .fill

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 أخر من البيانات في .fill مثل الـ char ينتج الـ ASCII value

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

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
65
```

وعند وضع Boolean value ينتج لـ 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

١- .front()

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
```

٢- .back()

تقوم باستدعاء أخر 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
```

٣- .at(element)

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
```

٤- .size()

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
```

٥- .empty()

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 ، User name، الـ Email، الـ About، الـ Address

ملحوظة: الـ String عبارة عن 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 متكونة من ١٢ حرف لان الـ string تنتهي بـ \0

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

**\0 => تقوم بـ Terminate الـ string أي انهائها**

```
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('\b') << "\n";      // \b => Backspace => ASCII Value => 8
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
0
8
```

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

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

**ملحوظة: الاختلاف الوحيد هو التخزين وسعته في الـ 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
```

١- Concatenating Strings: في علوم الكمبيوتر هي نظرية ربط الـ strings ببعضها 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 "include <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 append. أي اللحاق 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 علي حسب الفكرة

- Syntax الـ Loop With For:

```
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 i = 0 ثم هل الـ i < 6 إذن يقوم بطبعها وبعدها يقوم بزيادة قيمتها ١ ثم يقوم بطباعة ١ وهكذا حتي يصل إلي الـ ٥ ويقوم بطباعتها ويقوم بزيادة ١ ومن ثم الـ ٦ ليست أصغر من الـ ٦ إذن لا يقوم بطباعتها وتقف هنا الـ loop

## شكل مبسط لـ loop لفهم الفكرة

```cpp
int num = 0;
cout << num << "\n"; // 0
num++;
cout << num << "\n"; // 1
num++;
cout << num << "\n"; // 2
num++;
cout << num << "\n"; // 3
num++;
cout << num << "\n"; // 4
num++;
cout << num << "\n"; // 5
```

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
0
1
2
3
4
5
```

- تستخدم ايضاً لطباعة عناصر الـ Array

```cpp
int nums[4] = {100, 200, 300, 400};
cout << nums[0] << "\n";
cout << nums[1] << "\n";
cout << nums[2] << "\n";
cout << nums[3] << "\n";

for (int index = 0; index < 4; index++)
{
  cout << nums[index] << "\n";
}
```

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
100
200
300
400
100
200
300
400
```

## #048 - Loop With For - Advanced Syntax

### يمكن طباعة عناصر الـ Array هكذا

```cpp
int nums[] = {100, 200, 300, 400, 500, 600};
int numsCount = sizeof(nums) / sizeof(nums[0]); // 6*4 = 24/4 = 6

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

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
100
200
300
400
500
600
```

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

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

**ملحوظة٢:** من الممكن عدم كتابة الـ initialize, condition, update داخل الـ for وكتابتهم في مكان أخر.

### 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 للـ loop

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

ملحوظة٣: يجب مراعاة ترتيبهم الصحيح

```cpp
int nums[] = {100, 200, 300, 400, 500, 600};
int numsCount = sizeof(nums) / sizeof(nums[0]); // 6*4 = 24/4 = 6
int i = 0;

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

```
200
300
400
500
600
6
```

## #049 - Loop With For - Advanced Trainings

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

```cpp
int main()
{
  int nums[] = {100, 200, 300, 400, 500, 600};
  int numsSize = sizeof(nums) / sizeof(nums[0]);

  // 100, 300, 500 - Method 1
  for (int i = 0; i < numsSize; i += 2)
  {
    cout << nums[i] << "\n";
  }
  cout << "\n";

  // 100, 300, 500 - Method 2
  for (int i = 0; i < numsSize; i++)
  {
    cout << nums[i] << "\n";
    i++;
  }
```

```
100
300
500

100
300
500
```

- طريقتين لطباعة العناصر تنازلياً

```cpp
// 600, 500, 400, 300 - Method 1
for (int i = numsSize - 1; i > 1; i--)
{
  cout << nums[i] << "\n";
}
cout << "\n";

// 600, 500, 400, 300 - Method 2
for (int i = 5; i > 1; i--)
{
  cout << nums[i] << "\n";
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
600
500
400
300

600
500
400
300
```

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

```cpp
// Challenge
int i = numsSize - 1;
for (;;)
{
  i > 1;
  cout << nums[i] << "\n";
  i--;
  if(i == 1)
  {
    break;
  }
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
600
500
400
300
```

## #050 - Loop With For - Nested Loop

- الـ Nested loop: هو loop بداخله loop أخر مثل الـ If condition

```cpp
int main ()
{
  string products[] = {"Item 1", "Item 2", "Item 3"};
  string sizes[] = {"Small", "Large", "X-Large"};

  for (int i = 0; i < 3; i++)
  {
    cout << "Product Name:\n";
    cout << products[i] << "\n";
    cout << "Sizes:\n";
    for (int j = 0; j < 3; j++)
    {
      cout << sizes[j];
      if (j < 2)
      {
        cout << ", ";
      }
    }
    cout << "\n";
    cout << "========================\n";
  }
```

```
[Running] cd "f:\Programmin
++\#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 i
```

## #051 - Loop With While

- While الـ Syntax

```cpp
while (Condition Is True)
{

}
```

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

```cpp
int main()
{
  for (int i = 0; i < 5; i++)
  {
    cout << i << "\n";
  }

  int i = 0;

  while (i < 5)
  {
    cout << i << "\n";
    i++;
  }
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
0
1
2
3
4
```

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

- وبتغيير الترتيب تتغير النتيجة في الـ 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- الـ Do While: هو أن تقوم بتنفيذ الـ Block of code أولاً ثم التأكد من صحة الشرط عكس الـ while الـ Condition أولاً ثم تنفيذ الـ Block of code.

- Do While الـ Syntax

```
do
{

} while (Condition is True)
```

## مثالين لـ while

```cpp
int index = 4;

while (index < 6)
{
  cout << index << "\n";
  index++;
}
```

```
4
5
```

```cpp
int index = 6;

while (index < 6)
{
  cout << index << "\n";
  index++;
}
```

```
[Running] cd "f:\Programming\4- E
"f:\Programming\4- Elzero Web Sch

[Done] exited with code=0 in 0.32
```

## مثالين لـ do while

```cpp
int index = 4;

do
{
  cout << index << "\n";
  index++;
} while (index < 6);
```

```
4
5
```

```cpp
int index = 6;

do
{
  cout << index << "\n";
  index++;
} while (index < 6);
```

```
6
```

# #053 - Loop - Break, Continue

١- **Continue:** هو أن تقوم بعمل **skip** لـ **Element** داخل الـ **Array** عند عمل **for loop** لـ **Array**

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

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

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

```
Of Prog
Continu
10
After
30
After
40
After
50
After

[Done]
```

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

```cpp
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 Itration And Continue
  }
}
```

```
Contin
10
After
20
After
30
After
40
After
50
After
```

والدليل إذا قمنا بوضع cout الـ After بعد الـ Continue سيقوم بطباعة رقم 20 ولن يقوم بطباعة الـ After لأنه قام بعمل skip للـ iteration وهيا الـ After

```cpp
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 Itration And Continue
  }

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

```
"f:\Pr
Of Pro
Contin
10
After
20
30
After
40
After
50
After
```

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

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

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

```
Fun
Bre
"f:
Of
Con
10
30
40
50

[Do
```

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

```cpp
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]

[Runni
Fundam
Break,
"f:\Pr
Of Prog
Continu

[Done]
```

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

```cpp
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
Func
Brea
"f:\
Of P
Cont
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 عندم يوجد لديك عدد معين من الـ iterations تعرفه أي معروف عددهم**

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

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

### - Create Three Apps

### 1- Count Positive & Even Numbers Only

```cpp
int main()
{
  // Count Positive & Even Numbers Only
  int result = 0;
  int nums[] = {10, 20, -20, 13, 30, -30, 40};
  int numsSize = size(nums); // 7

  for(int i = 0; i < numsSize; i++)
  {
    if (nums[i] > 0 && nums[i] % 2 == 0)
    {
      result += nums[i];
    }
  }

  cout << "Result Is: " << result;
```

```
Result Is: 113
[Done] exited wi

[Running] cd "f:
With C++\#054 -
"f:\Programming\
++\#054 - Loop T
Result Is: 100
[Done] exited wi

[Running] cd "f:
With C++\#054 -
"f:\Programming\
++\#054 - Loop T
Result Is: 100
[Done] exited wi
```

## 2- Guess The Number

```cpp
// 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, Your Failed. The Number Is:" << guessNumbe
    break;
  }
}
```

## 3- Reversed Elements From User

```cpp
// Reversed Elements From User
int vals[5];
int inp;
cout << "Type 5 Numbers To Reverse\n";

// [0, 1, 2, 3, 4]

for (int i = 4; i > -1; i--)
{
  cin >> inp;
  vals[i] = inp;
}

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

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

## #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
```

- Function: هي Block of Code أي مجموعة من الأسطر البرمجية ننفذ 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. Built-In Function: وتسمي Standard Function ايضاً وهي موجودة في اللغة جاهزة ونقوم باستخدامها مباشرة.
٢. User Defined Function: هي الـ Function التي نقوم بإنشائها ونقوم بتسميتها الاسم الذي نريده ونقوم من خلالها بإرجاع نوع البيانات الذي نريده ونجعلها تقوم بتنفيذ المهمة الذي نريد تنفيذها.

٣- Syntax

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

ملحوظة: الـ Parameter هو معامل التجربة

٤- Declare A Function And Call It

```
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.
```

ملحوظة٢: void معناها فراغ أو فارغ

## #056 - Function With Parameter

- الـ parameter يوضع بين الـ square brackets وعند استدعاء الـ function نقوم بذكر اسم الـ parameter في الـ square brackets

```cpp
int main()
{
  sayHello()
```

```cpp
void sayHello(std::string
name)
```

**ملحوظة:** الذي يكتب في الـ function يسمي parameter ولكن الذي يكتب عند عمل call/invoke لـ function يسمي argument حيث يعتبر الـ parameter هو الـ variable والـ argument هو الـ value لـ variable

```cpp
void sayHello(string name)
{
  cout << "Hi " << name << ".\n";
}

int main()
{
  sayHello("Fady");
  cout << "Hi Ahmed.\n";
  cout << "Hi Sayed.\n";
  return 0;
}
```

```
Hi Fady.
Hi Ahmed.
Hi Sayed.
```

**ملحوظة٢:** عند وضع 2 parameters أو أكثر يجب أن تكتب كل قيمة من قيم الـ parameters في الـ arguments من نفس النوع.

```cpp
void sayHello(string msg, string name)
{
  cout << msg << " " << name << ".\n";
}

int main()
{
  sayHello(123, "Fady");
```

```
could not convert '123' from 'int' to 'std::string' {aka 'std::__cxx11::basic_string<char>'} gcc
(int)123
View Problem (Alt+F8)  No quick fixes available
```

**ملحوظة٣:** يجب ان يكون عدد الـ arguments عند استدعاء الـ function نفس عدد الـ parameters المكتوبين في الـ function

```cpp
void sayHello(string msg, string name)
{
  cout << msg << " " << name << ".\n";
}

in
{
  sayHello("Fady"); // Problem
```

```
too few arguments to function 'void sayHello(std::string, std::string)' gcc
void sayHello(std::string msg, std::string name)
View Problem (Alt+F8)  No quick fixes available
```

```cpp
void sayHello(string msg, string name)
{
  cout << msg << " " << name << ".\n";
}

in
{
  sayHello("Fady", "Ahmed", "Sayed"); // Problem
```

```
too many arguments to function 'void sayHello(std::string, std::string)' gcc
void sayHello(std::string msg, std::string name)
View Problem (Alt+F8)  No quick fixes available
```

# #057 - Function With Parameter Training

## مثال١:

```cpp
void iceBox(string item)
{
  if (item == "Coca Cola")
  {
    cout << item << " Will Be More Cold\n";
  }
  else if (item == "Apple" || item == "Juice")
  {
    cout << item << " Will Be More Fresh\n";
  }
  else
  {
    cout << item << " Is Invalid\n";
  }
}

int main()
{
  iceBox("Coca Cola");
  iceBox("Apple");
  iceBox("Juice");
  iceBox("TV Remote");
  return 0;
}
```

```
Coca Cola Will Be More Cold
Apple Will Be More Fresh
Juice Will Be More Fresh
TV Remote Is Invalid
```

```cpp
void iceBox(string item, string event)
{
  if (item == "Coca Cola")
  {
    cout << item << " " << event;
  }
  else if (item == "Apple" || item == "Juice")
  {
    cout << item << " " << event;
  }
  else
  {
    cout << item << " " << event;
  }
}

int main()
{
  iceBox("Coca Cola", "Will Be More Cold\n");
  iceBox("Apple", "Will Be More Fresh\n");
  iceBox("Juice", "Will Be More Fresh\n");
  iceBox("TV Remote", "Is Invalid");
  return 0;
}
```

```
Coca Cola Will Be More Cold
Apple Will Be More Fresh
Juice Will Be More Fresh
TV Remote Is Invalid
```

**الذي علي اليسار parameter واحد ولكن الذي علي اليمين بواسطة 2 parameter**

## مثال٢:

```cpp
void calc(int numOne, int numTwo, string op)
{
  if (op == "+")
  {
    cout << numOne << " + " << numTwo << " = ";
    cout << numOne + numTwo << "\n";
  }
  else if (op == "-")
  {
    cout << numOne << " - " << numTwo << " = ";
    cout << numOne - numTwo << "\n";
  }
  else if (op == "*")
  {
    cout << numOne << " * " << numTwo << " = ";
    cout << numOne * numTwo << "\n";
  }
  else if (op == "/")
  {
    cout << numOne << " / " << numTwo << " = ";
    cout << numOne / numTwo << "\n";
  }
}
```



```cpp
int main()
{
  calc(10, 20, "+");
  calc(40, 20, "-");
  calc(5, 10, "*");
  calc(50, 10, "/");
  return 0;
}
```

```
10 + 20 = 30
40 - 20 = 20
5 * 10 = 50
50 / 10 = 5
```

## #058 - Function Parameter Default Value

- Parameter Default Value: هو القيمة الافتراضية للـ parameter أي عندما لا يجد قيمة للـ parameter عند استدعاء الـ function يأخذ القيمة من القيمة الافتراضية

```cpp
void details(string msg = "Welcome", string name = "Unknown")
{
  cout << msg << " " << name << "\n";
}

int main()
{
  details("Hello", "Ahmed");
  details("Hi");
  details();
  return 0;
}
```

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

ملحوظة١: عند وضع Default Value للـ parameter الثاني لن يحدث error

```cpp
void details(string msg, string name = "Unknown")
{
  cout << msg << " " << name << "\n";
}

int main()
{
  details("Hello", "Ahmed");
  details("Hi");
  // details();
  return 0;
}
```

```
Hello Ahmed
Hi Unknown
```

ملحوظة٢: عند وضع Default Value للـ parameter الأول وعدم وضع Default Value للـ parameter الثاني يحدث error

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

```cpp
void details(string msg = "Welcome", string name)
{
  cout << msg << " " << name << "\n";
}

int main()
{
  details("Hello", "Ahmed");
  details("Hi");
  // details();
  return 0;
}
```

## #059 - Passing Array As Parameter

- Passing Array As a Parameter: حتي لا نجعل الـ function كـ static أي ثابتة حيث نقوم بالتعديل والجمع عند زيادة كل رقم هكذا لذا نستخدم الـ Array لكي نجعل الـ Function كـ Dynamic

```cpp
void calc(int n1, int n2, int n3)
{
  cout << n1 + n2 + n3 << "\n";
}

int main()
{
  calc(10, 20, 30);
  return 0;
}
```

```
60
```

- الـ Function الـ Dynamic بواسطة الـ Array

```cpp
void calc(int nums[], int count)
{
  int result = 0;
  for (int i = 0; i < count; i++)
  {
    result += nums[i];
  }
  cout << "Result Is: " << result << "\n";
}

int main()
{
  // calc(10, 20, 30);

  int arrayOfNumbers[] = {10, 20, 30 , 40, 100};
  int numsSize = size(arrayOfNumbers);

  calc(arrayOfNumbers, numsSize);
```

```
Result Is: 200
```

## #060 - Function - Return Statement + Void

١- No Return Function: تقوم بعمل action معين في الـ system فقط وهيا الـ void function

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

```cpp
void calc(int n1, int n2)
{
  cout << n1 + n2 << "\n";
}

int main()
{
  calc(10, 20);
  return 0;
}
```

```
30
```

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

void calc(int n1, int n2)
{
    cout << n1 + n2 << "\n";
}

int main()
{
    calc(10, 20);
    int result = calc(10, 20);
    return 0;
}
```

Error List
Entire Solution | 2 Errors | 0 Warnings | 0 Messages | Build + IntelliSense

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

**ملحوظة:** لا يمكن ان نقوم بعمل assign لقيمة الـ void function داخل variable ينتج خطأ من الـ compiler

**(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 أخر ثم نضغط = يقوم بإعادة قيمة لنا ومن الممكن أن نقوم بأخذ هذه القيمة وعمل له عملية أخري حيث يوجد قيمة حقيقة نتعامل معها.

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

```cpp
int calc(int n1, int n2)
{
  cout << "Operation Is Done\n";
  return n1 + n2;
}

int main()
{
  calc(10, 20);
  // int result = calc(10, 20);
  // cout << result * 5 << "\n";
  // cout << result + 20 << "\n";
  return 0;
}
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
Operation Is Done
```

```cpp
int calc(int n1, int n2)
{
  cout << "Operation Is Done\n";
  return n1 + n2;
}

int main()
{
  // calc(10, 20);
  int result = calc(10, 20);
  cout << result * 5 << "\n";
  cout << result + 20 << "\n";
  return 0;
}
```

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  {
```

٣- Nothing After Return: أي عملية او طباعة بعد الـ return لن تنفذ حيث أن الـ compiler عندما يصل إلي الـ return يقوم بالعودة لأعلي ليرجع شيء من الـ function 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
```

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

```
16  void calc(int n1, int n2)
17  {
18      cout <<
19      return 10;
20  }
    return-statement with a value, in function returning 'void' [-fpermissive] gcc
    View Problem (Alt+F8)  No quick fixes available
```

لكن عند وضع return; فقط لن يحدث error لأنها تعتبر void أي فارغة ونقوم بعمل الـ return; لأنها تقوم بعمل الـ break; أي قم بإنهاء 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 call/invoke لـ function قبل الإعلان - Declaration عن الـ function

```
using namespace std;
                'calc' was not declared in this scope; did you mean 'calloc'? gcc
int main()      int calc(int a, int b)
{               Declaration
  // Call/      View Problem (Alt+F8)  No quick fixes available
  cout << calc(10, 20);
  return 0;
}

// Declaration
int calc(int a, int b)
{
  return a + b;
}
```

٢- Forward Declaration: أي التسبيق بالإعلان عن الـ function قبل أن يتم عمل define/declare لـ function من الأساس ومن الممكن تكون في نفس الملف، ولكن في الأسفل أو في ملف أخر

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

int calc(int a, int b);

int main()
{
  // Call/Invoke
  cout << calc(10, 20);
  return 0;
}

// Declaration
int calc(int a, int b)
{
  return a + b;
}
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

30

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

int calc(int a, int b);

// Declaration
int calc(int a, int b)
{
  return a + b;
}

int main()
{
  // Call/Invoke
  cout << calc(10, 20);
  return 0;
}
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

30

المثال الذي علي اليسار لتصحيح الخطأ

لكن الذي علي اليمين لتوضيح أن الـ Forward Declaration لن يؤثر إذا كان الـ declare لـ function قبل استدعاء الـ function

**ملحوظة:** لكن لا يمكن الإعلان عن الـ function مرتين قبل أن نقوم بعمل call/invoke له ومرة أخري بعده سوف يحدث error وهو redefinition

```
// Declaration
int calc(int a, int b)
{
  return a + b;
}

int main()
{
  // Call/Invoke
  cout << calc(10, 20);
  re redefinition of 'int calc(int, int)' gcc
}    int calc(int a, int b)
     Declaration
// D View Problem (Alt+F8)  No quick fixes available
int calc(int a, int b)
{
  return a + b;
}
```

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

```
- Math Functions
--- pow
--- fmod
--- ceil
--- floor
--- round
--- trunc
```

**١- pow: وهي الأس أي power function**

```
cout << pow(2, 4) << "\n";      // 16
cout << 2 * 2 * 2 * 2 << "\n";  // 16
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
16
16
```

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

```
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 modulus function وهنا تكمن أهمية الـ function

```
cout << 11 % 
cout << 11.5 % 2 << "\n";        // Error
```

invalid operands of types 'double' and 'int' to binary 'operator%' gcc

View Problem (Alt+F8) No quick fixes available

٣- ceil: بمعني سقف وهي function لتقريب الرقم العشري وتحويله لعدد صحيح ولأنها بمعني سقف سوف يقوم بتقريبها للرقم الأكبر

```
25    cout << ceil(9.1) << "\n";      // 10
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
10
```

٤- floor: بمعني الأرض وهي function لتقريب الرقم العشري وتحويله لعدد صحيح ولأنها بمعني أرض سوف يقوم بتقريبها للرقم الأصغر

```
27    cout << floor(9.9) << "\n";      // 9
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
9
```

٥- round: يقوم بالتقريب حسب الرقم قريب من الرقم الأكبر أو الأقل حيث يختلف عن الـ ceil, floor مهما كانت الكسور سوف يقوموا بالتقريب 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 ٠,٥ أو أعلي من النص يقرب للرقم الأكبر ولكن إذا كان الرقم أقل من النص يقرب للرقم الأصغر

٦- trunc: وهي إزالة الكسور نهائياً وهي بمعني 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- tolower(): لجعل الحرف small ولكن ينتج قيمة الـ ascii value الخاصة به.

```
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- toupper(): لجعل الحرف capital ولكن ينتج قيمة الـ ascii value الخاصة به.

```
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
```

٣- ()isupper: للسؤال بـ is كما في الـ Boolean Expression تسأل إذا كان الحرف Capital

٤- ()islower: تسأل إذا كان الحرف small

مثال: Swap Case App

```cpp
// Swap Case App
string nameone = "ElZEro"; // eLzeRO eLzeRO
int nameoneSize = size(nameone);

for (int i = 0; i < nameoneSize; i++)
{
  if (isupper(nameone[i]))
  {
    cout << char(tolower(nameone[i]));
  }
  else
  {
    cout << char(toupper(nameone[i]));
  }

  // cout << nameone[i] << "\n";
  // cout << int(nameone[i]) << "\n";
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
eLzeRO
```

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

```cpp
// Remove Spaces App
string nametwo = "E\nl  z  \n\te r\t\no";
int nametwoSize = size(nametwo);

for (int i = 0; i < nametwoSize; i++)
{
  // if (nametwo[i] == ' ')
  // {
  //   continue;
  // }
  if (isspace(nametwo[i]))
  {
    continue;
  }
  cout << nametwo[i];
}
```

```
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 او الحرف الذي له ASCII Value أقل

```
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
```

→

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
-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 او الحرف الذي له ASCII Value أكبر

```
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
```

→

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
10
20
c
a
97
99
67
100
```

## Find Minimum Number App -٣

```cpp
// Find Minimum Number App
int nums[] = {10, -20, 30, -100, 100, -50};
int numsSize = size(nums);
int checkMinNum = 0;

for (int i = 0; i < numsSize; i++)
{
  if (nums[i] < checkMinNum)
  {
    checkMinNum = nums[i];
  }
}

cout << "Minimum Number Is " << checkMinNum << "\n";
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
-100
```

## Find Maximum Number App -٤

```cpp
// Find Maximum Number App
// int nums[] = {10, -20, 30, -100, 100, -50};
// int numsSize = size(nums);
int checkMaxNum = 0;

for (int i = 0; i < numsSize; i++)
{
  if (nums[i] > checkMaxNum)
  {
    checkMaxNum = nums[i];
  }
}

cout << "Maximum Number Is " << checkMaxNum << "\n";
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
Maximum Number Is 100
```

## Count Number Occurrence App -٥

```cpp
// Count Number Occurance App
int numsTwo[] = {10, 20, 10, 10, 13, 15, 100, 20, 10};
int numsTwoSize = size(numsTwo);
int counter = 0;
int choosenNum = 10;

for (int i = 0; i < numsTwoSize; i++)
{
  if (numsTwo[i] == choosenNum)
  {
    counter++;
  }
}

cout << choosenNum << " Found " << counter << " Times";
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
10 Found 4 Times
```

## #065 - Function Overloading

١- Function Overloading: وهو أن تقوم بعمل أكثر من function بنفس الأسم ولكن بإختلاف الـ parameters أو بإختلاف 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 تسمي overloaded function أي بنفس الأسم لكن عدد الـ parameters مختلف

ملحوظة: يحدث error عندما يكونوا نفس عدد الـ parameters ويخبرك بأن قومت بعمل redeclare للـ function

```
14  void print(int a, int b)
15  {
16    cout << "Number One Is: " << a << "\n";
17    cou redefinition of 'void print(int, int)' gcc
18  }       void print(int a, int b)
19          View Problem (Alt+F8)  Quick Fix... (Ctrl+.)
20  void print(int a, int b)
21  {
22    cout << "Number One Is: " << a << "\n";
23    cout << "Number Two Is: " << b << "\n";
24  }
```

ملحوظة٢: وعند استدعاء الـ function يبدأ بالـ function الأولي وعند وضع عدد parameters الـ function الثانية يقوم باستدعائها

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

→

```
t main()
                    void print(int a, int b,
print(10, 20);  2⁄2 int c)
print(100, 200, )
```

## ٢- function overloading مع تغير type الـ parameters

```cpp
void print(string a, string b)
{
  cout << "Text One Is: " << a << "\n";
  cout << "Text Two Is: " << b << "\n";
}

int main()
{
  print(10, 20);
  print(100, 200, 300);
  print("Fady", "Alamir");
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
Text One Is: Fady
Text Two Is: Alamir
```

```cpp
void print(string a, int b)
{
  cout << "Text One Is: " << a << "\n";
  cout << "Text Two Is: " << b << "\n";
}

int main()
{
  print(10, 20);
  print(100, 200, 300);
  print("Fady", 10);
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
Text One Is: Fady
Text Two Is: 10
```

## #066 - Function Recursion

**Function Recursion: هو عبارة عن technique من خلاله نجعل الـ function تـ call/invoke نفسها -**

```cpp
int add(int num)
{
  if (num == 0)
  {
    return 0;
  }
  cout << num << "\n";
  cout << "==================\n";
  return num + add(num - 1);
}

// 5 + (add(4))
// 5 + ( 4 + add(3) )
// 5 + ( 4 + ( 3 + add(2) ) )
// 5 + ( 4 + ( 3 + ( 2 + add(1) ) ) )
// 5 + ( 4 + ( 3 + ( 2 + ( 1 + add(0) ) ) ) )

int main()
{
  cout << add(5);
  return 0;
}
```

```
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- Vector: عبارة عن Container او حاوية نضيف 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 علي 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 Syntax لـ Vector ⟵ vector<type> VariableName

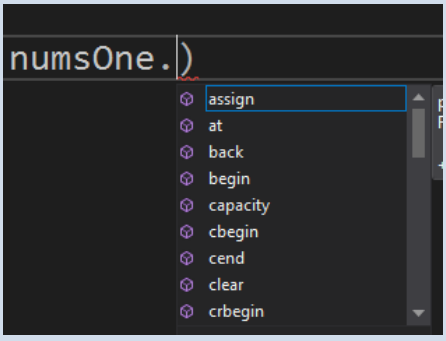
ملحوظة: في هذا المثال `vector<int> numsThree(4, 50);` قمنا بتحديد الـ size لكن نستطيع ايضاً ان نقوم بعمل Resize لها حيث أن الـ 4 هي الـ size أي ٤ عناصر والـ 50 أول قيمة في الـ Vector

يحدث error إذا قمنا باضافته لـ index[4]

```
numsThree[4] = 1000;
```

Expression: vector subscript out of range

ملحوظة٢: تظهر خصائص الـ class بعد الـ dot



## ٥- Loop on Elements

```cpp
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;
}
```

**ملحوظة٣: يمكن تعديل عنصر في الـ vector التي لا نستطيع التعديل عليها التي بـ parentheses بواسطة at هكذا**

**واضافة عنصر اضافي بواسطة خاصية الـ class وهي الـ push_back**

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

```cpp
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 | | |
|---|---|---|
| Array | Vector | |
| ١ | مكون من مكونات اللغة Language Construct<br>لا يحتاج إلى header file | يحتاج إلى الـ Standard Header File<br>#include <vector> |
| ٢ | لا نستطيع إضافة او حذف أي عنصر<br>أي لا نستطيع ان نقوم بعملية الـ Resize | بعد انشائه نستطيع إضافة/حذف عنصر<br>لأنه Dynamic ونستطيع ان نقوم بعمل Resize له |
| ٣ | العناصر في الـ Array هي Index Based<br>وذ access على العناصر عن طريق الـ index | العناصر نستطيع ان ذ access عليها عن طريق<br>الـ Iterator فقط لأن عناصر الـ vector not index based |
| ٤ | الـ Array أسرع من الـ Vectors | الـ Vector لأنه Dynamic فإن الدخول على العناصر<br>يكون أبطأ من الـ Array |
| ٥ | الـ Array تقوم بحجز مساحة أقل في الـ Memory<br>من الـ Vector | الـ Vector يقوم بحجز مساحة أكبر في الـ Memory<br>من الـ Array |
| ٦ | الـ Array توجد في لغة الـ C & C++ | الـ Vector موجود في لغة الـ 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
```

**ملحوظة١: عندما لا نعرف حجم البيانات الذي سوف نقوم بإضافتها نستخدم الـ Vector**

**مثل: مجموعة من المهارات يقوم باضافتها المستخدم ولا نعرف عددها لذا نستخدم الـ vector لوجود سهولة في إضافة البيانات او حذفها**

**ملحوظة٢: عندما نحتاج إلي السرعة والأداء نقوم باستخدام الـ Array**

**ملحوظة٣: نستطيع ان ننشئ Dynamic Array لكن الـ Vector أفضل**

**ملحوظة٤:** عندما نقوم بإضافة عنصر إلى الـ Array يحدث error لان 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";
```

```
Microsoft Visual Studio Debu
30
100
```

ملحوظة٥: وعند انشاء الـ Array بهذه الطريقة ايضاً 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";
```

```
30
100
```

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 الـ vector من الأفضل ان نقوم باستخدام at(). لأن الـ square brackets عندما نقوم باستدعاء عنصر غير موجود في الـ vector تعطينا garbage value لا نستطيع الاعتماد عليها او تركها في التطبيق

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

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
474
```

لكن الـ at() تنتج error وهو out of the range

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

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
terminate called after throwing an instance of 'std::out_of_range'
  what(): vector::_M_range_check: __n (which is 3) >= this->size() (which is 3)
```

٢- push_back: لاضافة عنصر في نهاية الـ vector

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

```
4
40
```

٣- Update: يمكن تحديث قيمة element في الـ vector عن طريقة الـ at().

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

```
40
100
```

٤- pop_back: تقوم بإزالة أخر عنصر من الـ 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 الـ vector الحالي

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

٢- max_size(): عدد العناصر التي تسطيع أن تقوم بإضافتها في الـ vector

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

```
1073741823
```

٣- capacity(): الـ storage المحجوزة لعناصر الـ vector واحياناً عددها يكون اكثر من عدد العناصر لأنها تعطينا الـ 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 الـ vector

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

```
10
```

٥- back(): تعطي أخر عنصر في الـ vector

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

```
80
```

ملحوظة: ونستطيع عن طريق الـ 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
```

٦- clear(): من خلالها نقوم بإزالة جميع عناصر الـ vector

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

```
0
```

٧- empty(): تستخدم مع الـ condition لنقوم بعمل check إذا كان الـ 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 Complecity 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
```

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

- لماذا نستخدم الـ Iterator:

١- يقوم بعمل simplify أي تبسيط للكود ولا نحتاج أن نري الـ Iterations كاملة الخاصة بال Container

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 Iterator بنفس الطريقة مع كل الـ containers

٥- يقلل نسبة التعقيد ووقت التنفيذ الخاص بالـ Application

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

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

nums.begin تشير إلي أول رقم في الـ vector `vector<int> nums = { 10, 20, 30, 40 };` وليس مجرد طباعة أي من بعد أن نشير له نستطيع أن نقوم بأشياء كثيرة مثل أحضار العنصر الذي بعده بعنصرين او حذف مجموعة العناصر الذي تليه لذا عندما يشير له يكون للـ iterator دور فعال.

**- طريقة أخري لإنشاء الـ iterator مباشرة:**

```
auto ite = nums.begin() + 1;

cout << "First Element Is: " << *it << "\n";
```

**ملحوظة:** عندما نريد طباعة ما يشير إليه الـ Iterator نضع قبله علامة الـ asterisk [*] هذه النجمة تقوم بعمل شيء يسمي 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
```

**ملحوظة٢:** وعندنا نعطيه nums.erase(0, 3) بالعناصر يعطينا error لأنه يريد الـ iterator `nums.erase(0, 3)`


لذا سوف نقوم بإعطائه الـ iterator هكذا

```
nums.erase(nums.begin(), nums.begin() + 2);

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

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

**ملحوظة٣:** أخر عنصر but not including `vector<int> nums = { 10, 20, 30, 40 };`
لذا يقوم بحذف أول عنصرين فقط

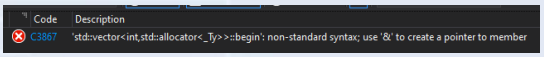
# #072 - Vector - Traversing With Iterator

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

- التنقل بين العناصر عن طريق الـ Iterator

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

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




## ١- begin()

```
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
```

## ٢- last()

```
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
```

**ملحوظة٢:** يجب وضع pranetheses حتي يشير إلي العنصر الذي يسبقه في الـ vector ولا يقوم بطرح واحد من الرقم الأخير بعد ان يشير إليه كما في المثال السابق

٣- advance: بمعني يتقدم وهو ان تقوم بالتقديم من عنصر لعنصر أخر داخل الـ 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
```

### ١- Loop With Iterator:

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

ملحوظة٢: من الأفضل ان نقوم بعمل pre increment لأنها أسرع لأنها لا تقوم بعمل copy لـ object وترجعه مباشرة `for (it = nums.begin(); it != nums.end(); ++it)`
ومن الممكن أن نقوم بعمل post increment ولكنه أبطأ

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

### ٢- Ranged Loop With For:

```cpp
// 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 ان نقوم باستدعاء الـ Header File الخاص بالـ algorithm لنستخدمهم

١- Count: هي function تقوم بحساب عدد مرات تكرار رقم في الـ 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.
```

ملحوظة٢: من المفضل أن نقوم بوضع علامة and [%] هنا لأنها تقوم بعمل access علي القيمة مباشرة لذا فإنها أسرع

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

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

٢- sort: هي function تقوم بترتيب عناصر الـ 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
```

٣- reverse: هي function تقوم بعكس ترتيب عناصر الـ 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
```

**- Pointers: معناها مؤشر وهو عبارة عن Variable يُخزن قيمة الـ Memory Address الخاص بـ Variable أو Object أخر.**

**ملحوظة: نقوم بوضع علامة الـ and [&] قبل اسم المتغير التي تسمي Reference Address Of أو Operator**

**مثل هذا المثال: نقوم بأخبار الـ compiler بأن هذا الـ pointer يساوي Memory Address of "num"** `int* ptr = &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
```

**ملحوظة٢: ولكي نصل إلى القيمة الموجودة في هذا الـ Address نقوم باستخدام الـ 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 الموجود داخل الـ Memory Address نقوم بالتعديل علي الـ Deference Operator Of Memory Address

ولن يتغير الـ Memory Address لكن تتغير القيمة

```cpp
int num = 100;
int* ptr = &num;

cout << "Value: " << num << "\n";
cout << "Address: " << &num << "\n";
cout << "Address: " << ptr << "\n";
cout << "Value: " << *ptr << "\n";

*ptr = 200;

cout << "Value: " << num << "\n";
cout << "Address: " << &num << "\n";
cout << "Address: " << ptr << "\n";
cout << "Value: " << *ptr << "\n";
```

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

## #076 - Pointers - Pointing To Array

- Pointing To Array: أن نشير إلي عنصر في الـ Array عن طريق الـ pointer

```cpp
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";
```

```
Elzero Web School\#2 - Fundamentals
Array\"app
First Element

Value With Index: 10
Value With Pointer: 10
Address With Index: 0x1705bffbc0
Address With Pointer: 0x1705bffbc0
================================
Second Element

Value With Index: 20
Value With Pointer: 20
Address With Index: 0x1705bffbc4
Address With Pointer: 0x1705bffbc4
================================
Third Element

Value With Index: 30
Value With Pointer: 30
Address With Index: 0x1705bffbc8
Address With Pointer: 0x1705bffbc8

[Done] exited with code=0 in 0.482 s

[Running] cd "f:\Programming\4- Elze
```

ملحوظة: في العنصر الثاني والثالث نقوم بوضع الـ pointer دخل prantheses لأننا عندما نقوم بعمل (ptr + 1) نخبره بأن يذهب لـ Memory Address الذي يليه ثم نقوم باستخدام الـ deference operator [*] علي القوس بالكامل لكي يُحضر القيمة الموجودة في هذا الـ Memory Address

ملحوظة٢: نوع البيانات integer والـ size of لـ integer يكون 4 Bytes

**ملحوظة٣:** عندما نقوم بوضع short تتحول من 4 Bytes إلي 2 Bytes

```cpp
short int nums[]{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
```

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

١- **Wild Pointer:** هو الـ pointer الذي لا يشير لـ Memory Address خاصة بمتغير أخر

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

```cpp
int *ptr1; // Wild

cout << ptr1 << "\n"; // Garbage Value
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
0x2860f9816b0
```

**ملحوظة٢:** وعندما لا نريد أن يعطينا قيمة Garbage Value نقوم بوضع NULL أو nullptr في القيمة ومن ثم يعطينا القيمة 0

```cpp
int *ptr1; // Wild
int *ptr2 = NULL;
int *ptr3 = nullptr;

cout << ptr1 << "\n"; // Garbage Value
cout << ptr2 << "\n"; // 0
cout << ptr3 << "\n"; // 0
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
0x83
0
0
```

**ملحوظة٣:** عندما نريد أن نوجه الـ pointer إلي بيانات لا نعرف نوعها سوف نقوم باستخدام الـ Void Array `void *ptr = &a;`

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

```
int a = 100;
void *ptr = &a;
cout << *ptr << "\n";
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 للـ pointer لنوع البيانات المناسب ويوجد نوعين من الـ Casting:

١- C-Style

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

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
100
```

٢- Modern

```
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: عندما نريد طباعة الـ element الأول والثاني في الـ Array نقوم باستخدام الـ square brackets أو الـ dereference operator

```cpp
cout << nums[0] << "\n";  // 1st Element => 10
cout << *nums << "\n";    // 1st Element => 10

cout << nums[1] << "\n";      // 2st Element => 20
cout << *(nums + 1) << "\n"; // 2st Element => 20
```

```
10
10
20
20
```

- Arithmetic Operator With Pointer

```cpp
int *ptr = nums;

cout << ptr << "\n"; // 1st Element => Memory Address
cout << *ptr << "\n"; // 1st Element => 10

ptr++;

cout << ptr << "\n"; // 2nd Element => Memory Address
cout << *ptr << "\n"; // 2nd Element => 20

ptr += 3;

cout << ptr << "\n"; // Last Element => Memory Address
cout << *ptr << "\n"; // Last Element => 50

ptr--;

cout << ptr << "\n"; // Before Last Element => Memory Address
cout << *ptr << "\n"; // Before Last Element => 40
```

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