ElCoM
ELECTRICAL COMPUTER MECHATRONICS

تقدم لجنة ElCoM الاكاديمية

دفتر لمادة:

برمجة الحاسوب

من شرح:

د. ميساء خزاعلة

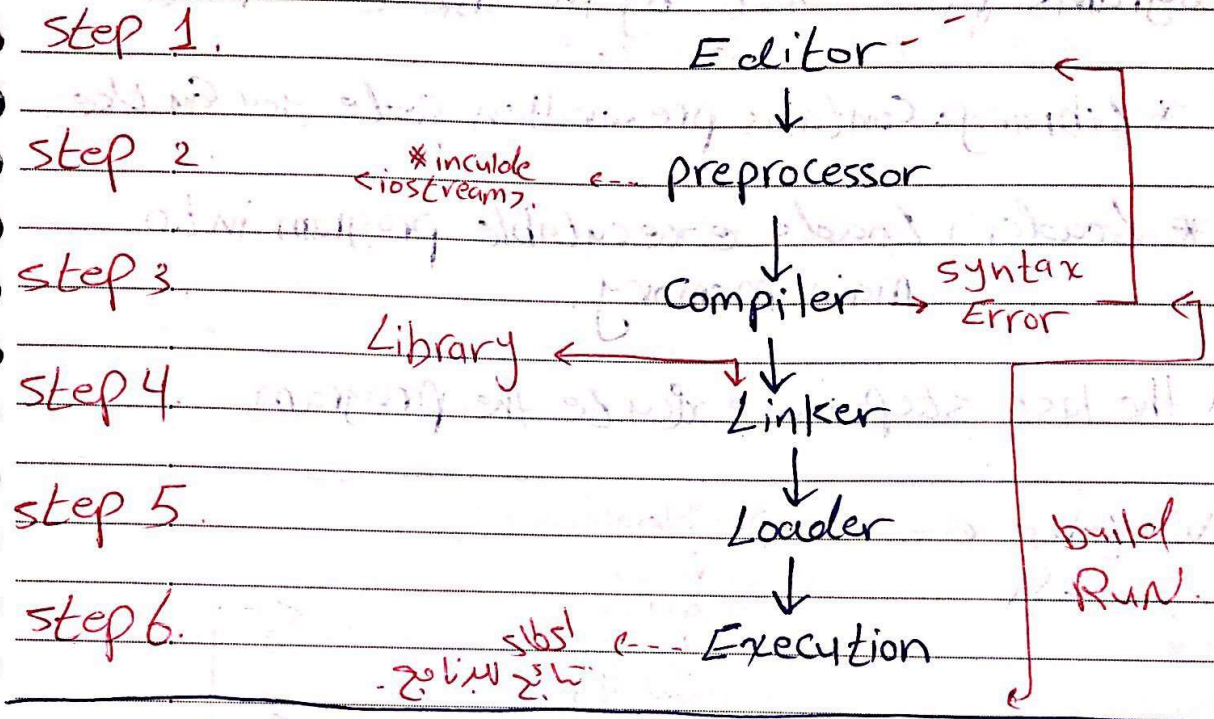
جزيل الشكر للطالب:

مؤمن القطامي

chapter "1".

كيف كتابة برنامج.

c++ program



⇒ Editor ---> source file .cpp.
⇒ Compiler ---> object file.
⇒ Linker ---> executable file.

⊛ types of Error:

[1] syntax Errors ---> أي مخالفة لقواعد اللغة.
[2] Logical Errors ---> طريقة حل المشكلة خاطئة.

⊛ preprocessor begin with #.

⊛ Use the Compiler to:

[1] check the program obeys the language rules.
[2] Translate into machine language "object program".

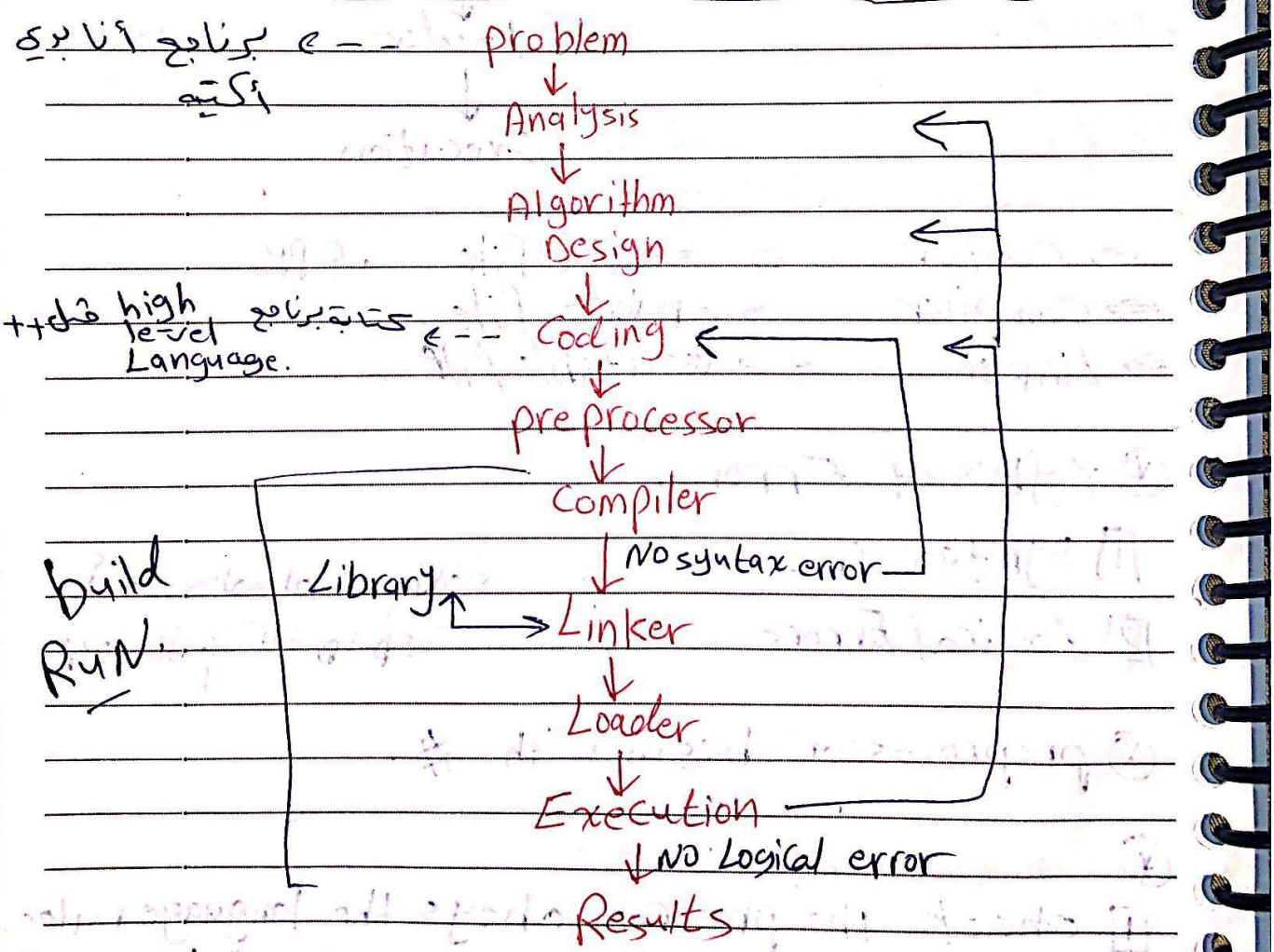
Scanned with CamScanner

⊛ Linker: Combines object program with other programs provided by the SDK to create "executable file"

* Library: Contains prewritten code you can use.

* Loader: Loads executable program into main memory.

* the last step is execute the program.



⊛ Maintenance

↳ صيانة أو معالجة لأي خلل للبرنامج.

Scanned with CamScanner

⊛ Programing with the problem

Analysis_Coding_Execution cycle.

[1]

⊛ Algorithm: - step by step problem-solving process.

[2] solution achieved in finite amount of time.

⊛ programing: is a process of problem solving

⊛ Algorithm: وانك تكتب خطوات الحل لمشكلة (هاي خول خطوة بعد تكتب مشكلة أو برنامج)

⊛ الخطوة الأولى هي "Analysis"؛

[1] step "1": Analyze the problem:

* out line the problem and it's requirements.
* Design steps (algorithm) to solve the problem.

"step 2": Implement the algorithm

[1] Implement the algorithm in code.

[2] Verify that the algorithm works.

⊛ step "3": Maintenance: program

- Use and modify the problem, if the problem domain changes.

⊛ افهم المشكلة (برنامج) واكون كل متطلباتها

⊛ واذا كانت المشكلة معقدة، حللها وجزئها، ولكل تكلم تحزيع algorithm.

- تأكد من صحة Algorithm

Can test Using sample Data.

some mathematical analysis might be required.

⊛ بإذا كانت لدينا Algorithm ، خريطة وصفية ، آلية Code
لغة عالية

- Enter the program Using text editor.

⊛ Run code through Compiler

- If Compiler generates error

[1] look at code and remove error.

[2] Run code again through Compiler.

⊛ If there are no syntax errors.

[1] Compiler generates equivalent machine code.

⊛ Linker links machine code with system resources.

⊛ Once Compiled and linked, Loader can place into main memory for execution.

⊛ the final step is to execute the program.

Scanned with CamScanner

Chapter "2" المحاضرة الثانية

Welcome To C++

```
// my first c++ program --> comment. single-Line
#include <iostream> --> preprocessor Directive.
using namespace std;
int main ( )
{
cout << "Welcome To C++ \n";
return 0;
}
```

Main function ← هون بنكتب الحل (البرنامج)

① // --- my first c++ --- ⇒ أي كلمة تبدأ هيك هاي كلمة تتم تجاهلها من قبل Compiler، هي عبارة عن ملاحظة لو كتبتها فقط ولبيت هل تنفيذية الـ Compiler بيشوفها بينزل ط

② include, using, int, return, cout ⇒ هدول عبارة عن كلمات محجوزة

③ ; ⇒ semicolon ⇒ فاصلة منقوطة / تعني نهاية الجملة

④ cout ⇒ هي دالة لـ print

⑤ \n ⇒ Back slash + n ⇒ new Line ط جديد

⊛ reserved word, keyword --> syntax error أي خطأ في الكتابة

⊛ cout << --> "insertion operator" هدول اسمهم

syntax error. لازم حاوط تكتبهم واذا ما كتبتهم

⊛ int, return --> مرتبطات مع بعض.

⊛ return 0 معناها البرنامج انتهى بشكل صحيح وتم تنفيذ كل الجمل ← terminate successfully

Scanned with CamScanner

* Small Litter (int, Using, include)
* white spaces ⇒ فراغات ⇒ مسافات يتجاهلها
ما يهمها تأثير على البرنامج, فممكن نحط فراغات
* C++ is sensitive Language
int ✓
Int ;
INT ✗

* طريقة ① لكتابة البرنامج

```
# include <iostream>
using namespace std ;
int main ( ) {cout << "welcome...\n" ; return 0 ;}
```

* طريقة ② لكتابة البرنامج

```
# include <iostream>
using namespace std ;
void main ( )
{ Cout << "welcome To C++ program \n" ;
}
```

* ملاحظات : ① في المثال اللي فوق ممنوع نحط return 0 بسبب syntax error, لما تحط int أو void

Scanned with CamScanner

الطريقة 4 لكتابة البرنامج:

```
#include <iostream>
using namespace std;
int main ( )
{
  cout << "welcome to c++ program" << endl;
}return 0
```

* ملاحظات: إذا أردنا كتابة // ، ماذا يطبع؟ single-line.

* إذا أردنا كتابة تعليق /* ——— */ ← "single line comment"

/* ——— */ → multiple Line Comment.

Ex:

```
#include <iostream>
using namespace std;
void main ( )
{ cout << "ABC\n";
  cout << 1+6 << endl;
(b) → cout << 7 >> "\n";
  Cout << "DEF";
  Cout << "10-4";
}
```

ABC
7
7
DEF10-4

* Escape sequence : ⇒ Cout فقط يستخدموا

\n → new Line	\a → alert	\b → back Space
\t → Horizental Tab	\" → print "	
\r → Carraige return	\\ → Print \	

Scanned with CamScanner

المحاضرة الثالثة:

Ex: ① cout<<"First\nExam"<<endl;
② cout<<"First\tExam"<<endl;
③ cout<<"First\bExam\n";
④ cout<<"First\rExam\n";
⑤ cout<<"First\\Exam"<<endl;
⑥ cout<<"\"First Exam\""<<endl;
⑦ cout<<"\'First Exam\'"<<endl;

Sol:

First
Exam ⇒ ①
First Exam ⇒ ②
FirsExam. ⇒ ③
Examt ⇒ ④
First\Exam. ⇒ ⑤
"First Exam" ⇒ ⑥
'First Exam' ⇒ ⑦

* ملاحظات كل الحالات:
1. ملاحظة كل الحالة الثالثة:
الواحد زيه زي الحرف عادية، وتاخد موقع.

2. ملاحظة كل الحالة الرابعة \r: يرجع لأول السطر to the begeining of the current Line.

3. ملاحظة كل الحالة الثالثة: الفاصلة \b ؛ هاي حابتقوم بحذف الحرف اللي قبلها، لا يكون من كلمة لحالها، أما واذا كان بين حرفين فيه أي حابتحذف الحرف قبلها، مثال.

Scanned with CamScanner

Ex: Cout << "c++\b\b"; | c++

Ex: Cout << "Real\b\b Green"; | R Green.

✱ ملاحظة: بالنسبة لـ \r ← الراصة تعمل return، overwrite، ترجع المؤشر لبداية السطر وتكتب الكلمة بدل الكلمة

Ex: Cout << "123\r";
Cout << "ab\n";
| ab3

Ex: Cout << "123\r" << endl;
Cout << "ab\n";
| 123
ab

✱ Token: [1] special symbol: +, -, { }, <, >, =
[2] Keyword: (int, return, void, using).
[3] Identifiers: name of variabel, class, function.

✱ Identifiers: variable name.

variable is a memory location.

- name
- value
- type
  - simple
  - pointer
  - structureal
- size

Scanned with CamScanner

✱ Variable name:
- Consist of only Letters ∫ Underscore(_) and numbers.
- must begin with letter or Underscore.
- predefiened Identifiers [cin, cout].

[1] Can't begin with numbers.
[2] Can't be keyword.
[3] No spaces (white spaces) ⇒ (space, tap, Enter).
[4] No special symbol expect (_).

Ex: Which of the following (legal, valid) identifier.
- First ✓
- 1st X ' --> يبدأ برقم.
- First = Exam X --> (_) مسموح استعمال هذا فقط
- First Exam X --> ممنوع يكون فيه فراغات.
- Int ✓
- int X --> ممنوع لأنها كلمة محجوزة

✱ ملاحظة: إحنا قلنا إنه غير مسموح (_) فيذا
كل اللي إنه ما تقله شوية.

- أي كل Capital: FirstExam
Firstexam ← "يزه" تكتب بس مكروه

✱ Variable Types:
[1] Integral: (int, charc, bool, Short, Long).
[2] floating point: (i.e. Decimal number), (float, double)
"Unsigned" ⇒ تخزن أرقام موجبة. Long

[1] int: عدد صحيح

- Size: 4 Bytes ⇒ stored Integer values ⊕ or ⊖

Ex: 1, -1, 10, -37.

---

[2] bool: 1 0

- size: 1 byte, store (true, false)
- Any non-zero value is true.

---

[3] charc:

size: 1 byte

- Store: Alphabet Letters, digits & any symbol.

A → Z
a → z
(0 → 9)

Note: value of charac closed within ' '

Ex: 'a', 'A', '+', '?', '0', '/', 'q'.

---

[4] float point numbers: أعداد حقيقية + صحيح

ملاحظة: الفرق بين float و double كالتالي:

1. float ⇒ size 4 Bytes.
2. double ⇒ size 8 Bytes.

Ex: 1,7 -1,88 3,4

ملاحظة: يجوز إنك تكتب float ← 10 ، بدل لو كان عدد صحيح ، ع يخزنها كـ شكل كسر عشري 10,0

Scanned with CamScanner

ملاحظة:- t \ ← هي مش زي الداتا بيز هي كده أحلى.

Variables Declaration:

معناها: عرّف أو اعلن، وكان لهاش memory بال Variable

- Type identifier و identifier أو

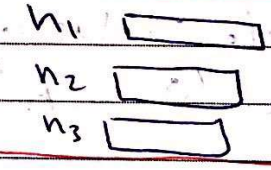
كل جملة بتنتهي بـ ; في نهاية كل جملة بال C++.

Exampels: Ex1 int x ؛ أعلن

معناها روح لـ x وكان في الذاكرة مقداره 4Bytes.

ملاحظة: قد يوجد قيمة ابتدائية للمتغير تسمى rubbish

Ex2: double n1 ، n2 ، n3 ؛



Ex3: char c ، float f ؛ x

هاي الجملة خاطئة لأنه، بين الأنواع المختلفة لازم أو لا يصير واني أفصلهم أو.

Ex4: int val ، Val ⇐ هاي الجملة صحيحة مية بالمية

لأنه تفرق بين Capital و small

Ex5: char E ؛ int E ؛ X

ملاحظة:- هاي الجملة خاطئة، ما يصير نعرّف أكثر من مرة، فقط مرة واحدة.

~~int or char تتعامل معهم Compiler لأنهم~~

Scanned with CamScanner

* Variable Initialization: قيمة أولية

Assignment operator (=) ⇒ variable = expression or value.

cin >>

Ex: int b ; b=5 ; ≡ int b=5 ;

* ملاحظة: لازم تعرف المتغير قبل ما تعطيه قيمة.

Ex: int x_1=30 ;
int x_2 ;
$x_2 = x_1 + 5$

→ Declaration.

x_1 [30]
x_2 [35]

ملاحظة: - لو كتبت هيك كده: ⇐ $x_1 + 5 = x_2$. X

Ex: char c_1 = 'a' و c_2 = '+' ؛

c_1 [a]
c_2 [+]

* ملاحظة: لو نسيت هدول ' ' ' ' syntax error

* ملاحظة: - أي char value، لازم تحطهم بين ' '.
* ملاحظة: - من الأمثلة السابقة لو كتبت [a] بدل a طارح ليرن c_2 شو هيه.

Ex:- float f_1=5 , f_2 ؛
$f_2 = f_1$ ؛
$f_2 = f_1 + 10$ ؛

f_1 [5.0]
f_2 [5 15.0]

* التغيير تبع القيمة عادي هيك.

~~Logical error~~

* ملاحظة: - من الأمثلة السابقة لو كتبت $f_1 = f_2$ ⇐ $f_1 f_2$ معه قيمة rubbish →

Scanned with CamScanner

[2] طريقة for variable Initialization:

⊛ Initialize Using cin >> → Extraction operator.

Ex: int v;
cin >> v;
what is the value of v ? لو User دخل

⊛ ملاحظة: لو دخل syntax error

⊛ ملاحظة: ممكن نعمل prompt User with message

Ex: int v;
cout << "Enter integer value\n";
cin >> v;

User دخل: 5

```
Enter integer value
5
```

Ex: float f1, f2;

cin >> f1;
cin >> f2; ≡ cin >> f1 >> f2;

cin >> f1, f2; ⇒ Logical error

~~Rubbish ← f2 لو دخل f1 بس~~

⊛ Constant variables:

memory location their values can't be changed.

⊛ syntax:

const type identifier = value ;

Ex: const float pi = 3.14

Ex: const float pi = 3.20 ⇒ syntax error (محاولة لتغيير قيمة ثابت)
cin >> pi ;

Ex: const float pi ;
pi = 3.14 ; ← هاي خطأ، لازم أعطيه قيمة

الـ declaration is Initialization
يعني: constant لازم كلهم بنفس السطر.

⊛ C++ operators :-

1) Arithmatic operator in C++ :- العمليات الحسابية

- Divide: /
- multiply: *
- ADD: +
- subtract: -
- modules: %

} inter float data types

⊛ priority ⇒ "order of precedence"

1. ( )
2. / * % Left to right.
3. + - Left to right.

Scanned with CamScanner

Ex: Cout << 2+8/4 ;
Cout << 5.3 - 2*2 ;

4
1.3

Ex: int x = 5.3 - 2*2 ; ⇒

Cout << x ;

1

Logical error

لم يأتي من باحث

Ex: double x = 5.3 - 2*2 ;
Cout << x ;

1.3

Ex: bool x = 5.3 - 2*2 ;
Cout << x ;

1

الأقواس بالخانة

* float or double:

- Remark: $20E02 = 20 * 10^{2} = 2000$

$1.75E3 = 1.75 * 10^{3} = 1750$

$50E-2 = 50 * 10^{-2} = \frac{50}{100} = 0,5$

* ملاحظة: الأرقام فوق بأنه إذا كتبت E أو e مضاعفوة (E كاتبال أو صغيرة)

Ex: float f1 = 1.5E3 ;
float f2 = 75e-2 ;
Cout << f1 << "\n" << f2 ;

1500
0,75

Scanned with CamScanner

⊛ Charc: single ' ' بين Charc لازم يكون ال variable اللي هتخزن فيها

Ex: char c1 = 'A';

⊛ Ascii Code:

⊛ ملحوظة مهمة: كل Charc بخزن او بدلالة "Ascii code" ليها رقم معين وهو بيكون integer value

A Ascii→ 65	a Ascii→ 97	0→48
B Ascii→ 66	b Ascii→ 98	1 Ascii→ 49
C Ascii→ 67	c Ascii→ 99	2 Ascii→ 50
:	d Ascii→ 100	
z → 96		

↑ جدول ⊛ ↑ لازم تكون كابيتال ↑

⊛ Division operator /:

- int / int = int

Ex: 5 / 2 = 2

- float / int = float

Ex: 5.0 / 2 = 2.5

- int / float = float

Ex: 5 / 2.0 = 2.5

- float / float = float

Ex: 5.0 / 2.0 = 2.5

Scanned with CamScanner

* Modulus: % ⇒ معناها باقي القسمة

5 mod 2 ⟵ كيف نكتبها في البرمجة] ⇒ 5 % 2 = 1

الناتج هو الجواب

Ex: 13 % 3 = 1

5 % 10 = 5
10 % 5 = 0
] ⇒ ملاحظة: في بين القسمة وباقي القسمة.

* ملاحظة: خارج باقي القسمة % لا يأتي إلا بين عددين يكونوا من نوع Integer.

---

* Casting:

معنى الـ Casting: هو تحويل نوع data من نوع type إلى نوع type آخر.

[1] Implicit :- Done by Compiler.
[2] Explicit :- Done by User.
→ C-style
→ static_cast.

---

[1] Implicit: التحويل يتم من قبل الـ Compiler في الذاكرة.

Ex: int x = 10.3 ؛ → 10
int x2 = 11.7 ؛ → 11
int x3 = 'B' ؛ → 66 ← ascii code
int x4 = true ؛ → 1

Casting.

* عملية ادخال الحرف أو bool أو الرقم الصحيح كما هي في مثال الـ charc

هيك بتخزن في الذاكرة.

4,0

```
Ex: float f1 = 4;
    Char c = 100
    Cout << c << c+1;
```

d 101

* ملاحظة: هيز كتر فتح بين الطباعة وشو بتخزن، الـ Char فوق هو عبارة عن ascii، بيكي اطلع المقابل له وأطبعه.

* لاحظ الحالة الثانية لـ Char دخل لهيك حالة بتوقعي فيه من خلال الـ ascii له.

```
Ex: char c1 = 'a';
    Char c2 = c1 + 2;
    Cout << c1 + c2;
```

98

* لما كتب إنه الصفر هون هو عبارة عن Char وليس رقم، وإذا بدي أدخل لهيك حالة باظبطه هو، لازم بجيب الـ ascii له.

* وإذا شفت الـ Char بين ' ' وبدي أقارنه بجيب الـ ascii.

```
Ex: Char c = '0';
    Cout << (100/c);
    Cout << endl << c;
```

2
0

* ملاحظة: - إذا الأرقام كانت بين ' ' هون الـ Char تعامل معاها داخل الهيلات لحالة بالـ ascii Code.

* الحالة الثانية طبعت الـ Char نفسه لأنه عادي كلمة حالة.

* ~~ملاحظة: لو الصفر فوق عادي بين Single Quote ← syntax error~~

Scanned with CamScanner

1 byte

Char: نوع بيانات حرفية ← 1 byte

(' ') ascii code ← 8 بتات تطبيعي تخزين

A ← 65 الرقم قبل الـ

* ملاحظة: أي شيء بين ' ' يعتبر char
* أي شيء بين " " يعتبر syntax error

Ex: Char c=98;
cout<<c;

b

[2] Explicit casting:

Ex: int a=5, b=2;
cout<<(float(a)/b);

2.5

syntax for Explicit casting: (الصيغة)

static_cast <new type> (+)

↳ : تحويل

Ex: int a=5, b=2;
Cout<<(static_cast<float>(a)/b);

2.5

* ملاحظة: إذا قلنا سوف تحويل مؤقت للقيمة الحالية في int وليس float، ولكن يخزن القيمة/نسبة.

a=97 ↳ int x=|a| ، int تبقى

* واذا بدل تخزن في نوع int

Ex: Char c = 'B';
cout << static_cast<int>(c);
cout << endL;
cout << c;

66
B

* ملاحظة: في الكود الأول، أنا حولتها من Char إلى int بشكل مؤقت، أما الكود الأخير فهي زي هو.

* bool b = '0'; b [1]
bool b = 0; b [0]

الحالة الثانية:

Ex: int a=9, b=2;
cout << a/b;
cout << static_cast<double> a/b;
cout << endL;
cout << static_cast<float>(a+0,5);
cout << a;

44.5
9.59

* لاحظ كيف إنه لهون في السؤال بس إنه تعليمة الـ Cast هي كبيرة كتير تحويل مؤقت للقيم داخل العمليات الحسابية، وبعدها رجع القيمة الأصلية

* انتبه لهيك وإذا فيه Const أو لأ ممكن يكون بالواجبات.

ملاحظة: إذا خزنت int x='A' ← الكمبيوتر يخزن A=65 في الـ char code وهو رقم يمثل الحرف A حسب جدول الـ ASCII.

Increment & Decrement operators:

- Unary operator

post	pre
x++	++x
x--	--x

← يتبع بعد | ⇒ يتبع قبل

بحيث: تابع أو يظهر القيمة الإضافية.

ملاحظة: post و pre فقط في الـ expression، يفرقوا عن بعض في حالة الـ

Ex: int x=10, y=10;

```
x++; // post increment
cout<<x<<endl;
++y; // pre increment
cout<<y;
```

11	
11	

ملاحظة: كيف فرق انو x, y هون تماماً واحد طلعوا بهالـ output؟ لأنه كلاهما في حالة (واحد طلعوا بالـ expression).

Ex: int x,y,a=5,b=5;

```
x= ++a;
y= b++;
cout<<x<<"\t"<<y<<endl;
cout<<a<<"\t"<<b<<endl;
```

6	5
6	6

ملاحظة: كيف انه هون الـ x و y واحد طلعوا نفس الـ expression؟ لأنه كلاهما ثابتة أو نفس القيمة لا تتغير في المثال السابق.

Scanned with CamScanner

* في هاي الحالة، اللي اختلف معانا شيئا هو المتغير b، لأنه وزدنا قيمته الأساسية وزدنا عليها واحد، يعني (b++) = 2 هاي معناها بتأخذ القيمة الابتدائية لـ b وهي 1 من أول تعبير أو جملة طباعة وبعدها بتزيد القيمة b واحد في الجملة الثانية وهي الطباعة الثانية.

* زيادة مقدار واحد، فاصلة عامة لازم تحتها فاصلة منقوطة ;

* ملاحظة: إذا واضطيت int قبل المتغير ← syntax error

Ex: int a=7 , b=9 ;
int c=(++a)+(b--);
Cout<<a<<" "<<b<<" "<<c;
Cout<<"\n";
Cout<<++a<<"\n";
Cout<<a;

8 8 17
9
9

* ملاحظة: في أول عملية طباعة فقط تأخذ القيمة الابتدائية b--. وبعدها تأخذ تعديل القيمة بمقدار واحد.

* في أول a++ ← ازاد لـ a واحد كل طول وفي الأمر الثاني ازاد كمان واحد فا صار a=9.

* مثل السؤال السابق لو كانت الجملة قبل الأخيرة هيك
Cout<<a++<<"\n";

~~كان طبع 7 8 8 بتأخذ أول~~
قيمة بعدها بعدل
8
9

Scanned with CamScanner

Ex: float a=10.1;
float b=7.4;
int c=(a--)-(b--);
cout<<a<<b<<c;

9.16.42
a b c

* أول expression هو int c, تأخذ القيمة في الطباعة كما هي ثم يتم أول تغيير (التغيير داخل الأقواس).

* وظيفة: Run time Logical error ← $\frac{100}{zero}$

* +=, -=, *=, /=, %=

Ex: a=a+5 = a+=5
x=x%3 = x%=3

* String Data type.

```
#include <string>
#include <iostream>
using namespace std;
void main( )
{
string s1="welcome";
string s2="To c++";
cout<<s1<<s2;
}
```

welcomeTo c++

Scanned with CamScanner

User input Welcome To C++

Ex: string s;
cin >> s;
cout << s;

ما بخزن داخل الكلمة.

Welcome To C++
welcome

(*) أكتب ٣ طرق مختلفة لـ x=x+1 ⇐ sol:

[1] x+=1; [2] x++; [3] ++x

الطريقة الرابعة:

ملاحظة: تحويلة من string إلى string على نوع string عادية لقدرتها على تخزين أكثر من char فما لو كان أكثر من char واحدة، شرط الـ char يكون بين " ".

char x = 'a';
string s = "a"; ← syntax error ← ' ' و ' '
Initialize by " " ↵

(*)

Ex: string s1 = "HC All";
string s2;
cin >> s2;
cout << s1 << endl << s2;

cin >> s1;
cout << endl;
cout << s1 << endl << s2;

الـ user ما بتخزن.

HC All
HC All
HC
All
HC

Given that user input for s2 is "HC All".

(*) تخزين All لحالها وما لـ HC

Scanned with CamScanner

* ملاحظات على المثال السابق:-

* ال cin مرة، string، بتستنى تاخد لحد أول space.
* لاحظ كيف أنه مرة cin ثانية لـ s1، فبتروح اتقفل لـ User يكتب تاني مرة تانية، وتاخد اللي بعد أول فراغ تلقائيا.
* s2 مرة، واتقفلت تلقع Hi، لأن cin بتاخد لحد أول فراغ.

Ex: string s1;
getline(cin, s1);
cout << s1;

User input ⇒ Hi AH

Hi AH

* ملاحظة: الـ get line ⇐ فايدتها انها تاخد كلمة من أول لـ b لآخره وما تهتم للفراغات.

Ex: char c = ' ';

* استخدام cin بيطلب أكتر من فراغ، لازم أكتب تاني.

Ex: int x
cin >> x;
cout << x;

لو دخلت حرف
input failure
وناخد int.

cin ⇒ A :لو دخل

ما بيعمل Casting، بيخزن Rubbish أو Zero.

Scanned with CamScanner

3-3

Ex: Suppose you have the following variable declarations

```
int a, b;
double z;
Char ch, ch1, ch2.
```

⇒ the following statements show how the extraction operator >> works.

statement	Input	value stored in memory
1) `cin>>z>>ch>>a;`	36.78B34	z=36.78, ch='B', a=34
2) `cin>>z>>ch>>a;`	36.78 B34	z=36.78, ch='B' a=34

بالنسبة لـ float بالنسبة لـ nondigit ← الواقع ما اختلف.

3) `cin>>a>>b>>z;`	11 34	a=11, b=34 Computer waits for the next number

* لاحظ كيف بالنسبة لـ int بالنسبة لعدد أو رقم أو لعدد ثابت رقم.

4) `cin>>a>>z;`	78.49	a=78, z=0.49

* تأكد تماماً بالنسبة لـ cin ليست Casting، فقد ثابت نوع.

5) `cin>>ch>>a;`	256	ch='2', a=56

* كما تتوقع بالنسبة لـ char هو أي ثابت من مزلة وحرف، فقد 2 int، وباقي Charc.

6) `cin>>a>>ch;`	256	a=256, Computer waits for the input value of ch.
7) `cin>>ch1>>ch2;`	A B	ch1='A', ch2='B'

لوقف قراءات نفس الأشياء.

Scanned with CamScanner

Example "3-2"/

suppose you have the following variable declarations

```
int a;
double z;
char ch;
```

⇒ The following statements show how the extraction operator >> works.

	Statement	Input	Value stored ...
[1]	`cin>>a>>ch>>z;`	57 A 26.9	a=57, ch='A', z=26.9
[2]	`cin>>a>>ch>>z;`	57 A 26.9	a=57, ch='A' z=26.9
[3]	`cin>>a>>ch>>z;`	57 A 26.9	a=57, ch='A' z=26.9

✗ لا تنظر Enter ولا تتوقف هنا.

[4]	`cin>>a>>ch>>z;`	57A26.9	a=57, ch='A' z=26.9

✗ لا تنظر int تتوقف لأن أول خانة digit.

✗ char مثلك.

✗ لعند أول فراغ، أو أول enter ~~enter~~.

Scanned with CamScanner

Ex: (3-1)

Suppose you have the following variables declaration

```
int a,b;
double z;
char ch;
```

⇒ the following statements show how the extraction operator >> works.

	Statement	Input	Value stored
1	cin>>ch;	A	ch = 'A'
2	cin>>ch;	AB	ch = 'A', 'B' is held for later input.
3	cin>>a;	48	a = 48
4	cin>>a;	46.35	a = 46, 0.35 is held for later input.
	Non-digit number: . أسئلة، في كل		
5	cin>>z;	74.35	z = 74.35.
6	cin>>z;	39	z = 39.0
7	cin>>z>>a;	65.78 38	z = 65.78, a = 38

Scanned with CamScanner

⊛ الخوارزميات السابقة .

⊛ تذكير : كنا بنقول إنه في حالة float و double و cin ، بيوقف عند أول فراغ أو أول non-digit

⊛ Ex: char x = 97 ; ascii code → x [a]

= assigment operator بتاعة بتخزن

⊛ Ex: Char x

cin >> x ; input → a → x [a]

لاحظ إن assigment بتختلف عن cin ، assigment بتخزن و cin بتخزن

Ex: int x = 'a' ; cout << x ; [97]

لو خزنت x = a هيطلع syntax error

Ex: int x ;
cin x ; input → a

هيطلع Zero

لأن int وظيفتها Char ، لما أنا دخلت بالغلط حرف بدل رقم بيطلع Input failure و بيطبع الحالة 0

⊛ تذكر إنه في Casting بالنسبة لـ cin

⊛ predefind functions :

1. pow(x,y) ⇒ x^y
2. SQRT(x) ⇒ $\sqrt[2]{x}$

arguments عدد الـ

⊛ arguments: هي القيم التي يحتاجها الـ function وهي من نوع double (بالأغلب بتكون).

⊛ أما بالنسبة لـ result ← return value of type double.
(كل هاد هون كنز بس لتوقف فيه بـ chapter functions)

⊛ كمان أقرب شغلة هدول لازم نأكل: predefined functions

※ include <cmath>.

* لازم نعملها لأنه هدول الفانكشنز (pow, SQRT) معرفات كليها بس، إذا ما عملتها بيطلع syntax error.

Ex:
```
※include <cmath>
※include <iostream>
Using namespace std;
void main ( )
{ int a=sqrt(24);
  float b=pow(5,2);
  cout <<a <<"\t"<<b ;
```

4 25.0

أو 25 نفس الشي.

⊛ Chapter "4":

Control structure.

- Selection (if, if else, switch) ← Ch.4
- Repetition [for, while, do while] ← ch.5.

Scanned with CamScanner

⊛ Relational operators:

- `>` ---> greater than
- `>=` ---> greater than or equal to.
- `<` ---> less than
- `<=` ---> less than or equal to.

هدول بسموهم Binary operators ← لإنك قاعد تقارن بين طرفين.

⊛ Equality operators:

1. `==` equal
   `!=` not equal
   } same priority ⇒ الأولوية من Left to Right.

⊛ Logical operators:

- `!` → Not [1]
- `&&` → AND [2]
- `||` → OR [3]

هاي بوقف أكتر شوي ^^

الأولوية من فوق لتحت.

⊛ في حالة تساوي Left to Right.

⊛ order of precedance:

0 → false
1 → True

A	B	A && B	A || B
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Scanned with CamScanner

(X) الأولوية بالترتيب:

1. ( ) , ++, --.
2. !
3. / % *
   + -
4. < <= > >=
5. = !=
6. &&
7. ||
8. +=, -=, *=, /=

Ex: `Cout << !('A' > 'B');`
`Cout << !(6<=7);`

```
1
0
```

Ex: `Cout << (14>=5) && ('A'<'B')`
(14>=5) true, ('A'<'B') true

```
1
```

Ex: `Cout << (24>=35) || (7!=7);`
(24>=35) false, (7!=7) false.

```
0
```

Ex:
```
int a=5, b=2;
bool b1;
b1=!(a>2);
Cout << b1;
Cout << (a>b) && (b>=1);
Cout << (a<b) && (b>=1);
Cout << (a<b) || (b<=1);
```

```
0 1 0 0
```

Scanned with CamScanner

Ex: Write a program a Complet C++ program that print the Ascii Code for any Charachter input by the User {

Sol.

```
#include <iostream>
using namespace std;
void main ( )
{ char x;
  cin>> x;
  cout<<static_cast<int>(x);
}
```

او

```
#include <iostream>
using namespace std;
void main ( )
{
char x;
cin>> x;
cout<< (int(x)); }
```

Scanned with CamScanner

المحاضرة التاسعة

مقارنات جمل string

* Suppose we have the following declarations:

```
string str1 = "Hello";
string str2 = "Hi";
string str3 = "Air";
string str4 = "Bill";
string str5 = "Big";
```

ascii حسب

str1 < str2 ---> e < i True

* لاحظ كانت تشابه بحرف H، وشيناه، وقارنا اللي بعده.

str1 > "Hen" l < n false

str3 < "An" i < n true

str1 == "hello" H < h false

* لاحظ أن الأحرف الكابتل < السمول.

str3 <= str4 A < B true

str2 > str4 H > B true

str4 >= "Billy" false

* ملاحظة: الكلمات اللي مفهم تطابق في جميع الحروف لكن وحدة منهم تحتوي حروف أكثر من الثانية، هيه اللي تكون أكبر.

str5 <= "Bigger" true ---> عدد حروفها أكثر.

* تذكر: لازم تنزل #include<string> ... قبل منذ فاهم.

* كيف نوف الحرف هاذا أكبر من الحرف الثاني؟ من خلال ascii

* دائماً الحرف الصغير أكبر من الحرف الكبير.

* في حال تساوي أو تطابق حروف كلمتين، شوف مين عدد حروفه أكثر من الثاني.

* اللي فوق كان عبارة عن مقارنات جملة داخل string.

(If)

* There are 3 types of If clause:-
  1. If ---> one way selection.
  2. If-else ---> two way selection.
  3. Switch ---> Multiple selections.

---

1. if ⇒ (syntax):

if ⇔ if (expression) statement;

* ملاحظة مهمة: لا توضع بعد statement ← semicolon ; وليس بعد جملة if.

* إذا كانت جملة if ← true ← ينفذ الجملة التي بعدها إلى نهاية الـ if، وإذا كانت false أو كانها موجودة.

Ex: int x = 6, y = 7;

```
if(x++<y)
cout<<"Hi";
cout<<endl;
cout<<x<<"\t"<<y;
```

Hi
7 7

* لاحظ كيف إنه الجملة الموجودة بعد الـ If هي الجملة الوحيدة التابعة لها وانهيتها بـ ;.

* تذكير معجزة: الـ Increament و Decrement إذا دخلوا في expression "أو أي عملية حسابية" يستخدم القيمة الابتدائية ويعرها بعد القيمة الـ Memory ~~في حالة كانت Pre~~ Post

Scanned with CamScanner

Ex: int x=6, y=7 ;
if (++x < 7)
Cout << "Hi" ;
Cout << endL ;
cout << x << "\t" << y ;

7	7

* هون كنا pre ← كاطول أولوية كالقيمة.

Ex: char c_1='A' , c_2='b' ;
if (c_1 >= c_2)
{ c_1++ ;
c_2-- ;
}
Cout << c_1 << "\t" << c_2 ;

A	b

Ex: نفس المثال السابق بس:
c_1='k', c_2='B'

F	A

* طبعاً كان بدنا نحكي لأنه لما كان ++c, c-- ما رح يطلعوا قوة بكلمات حياتي، لو رح يطلعوا كنت طلعت ascii

* لو سويناهم pre بدل post الكل نفس القوة.

Scanned with CamScanner

* ملاحظة :- في الأمثلة السابقة إذا دخلت جملة أكثر من جملة واحدة فإنه يجب أن تكون هذه الجمل وكذلك القيم الخارجة بين قوسين { } (braces).

Ex: Char c_1 = 'A';
Char c_2 = 'b';
if (c_1 >= c_2)
c_1++;
c_2--;
cout << c_1 << "\t" << c_2;

A a

* لاحظ كيف أن c_1++ هي فقط التي تابعة للجملة if لأنها الوحيدة بدون أقواس.

* تذكير أخوك ^ :
الـ pre و post ، (Increment) و (decrement) ، وإذا كانوا في جمل متتالية لبعض كما طول ، أما إذا دخلوا في expression هون بتتغير post ، بتتخزن قيمة أول بمرة لبعض بالـ Memory .

Ex: write a c++ program to print odd if the user input is odd number otherwize print even ?

Scanned with CamScanner

User

```
#include<iostream>
Using namespace std;
void main ( )
{ int val;
Cout<<" Enter integer Nu: ";
Cin>>val;
If(val%2==0)
Cout<<val<<": Even\n";
If(val%2!=0)
Cout<<val<<": odd\n";
}
```

```
Enter integer Nu: 5
5: odd
```

* في الحالة ⑦ لازم نكتب == ، وكتابة = تعبير عن assignment

Ex:
```
int a=2, b=4;
if (a==2)
Cout<<"AA";
Cout<<a<<"\t"<<b;
```

0 4

هون فعلياً في قيمة لـ a.

* في الحالة السابقة a=a-2 معناها a-=2 وقيمة هي 0 ، و If(0) false لن ينفذ لهيك.

If (أي شي قيمته غير 0) ⇒ true. ⇒ ينفذ.

Scanned with CamScanner

Ex: int a=4, b=4;
if (a--=2)
Cout << "AA";
Cout << a << "\t" << b;

AA2 4

* if(5) → true
if(0,000000001) → true
if(!7) → false
if(!'+') → false
⇒ if 7 → true → if(Not true) → false

Ex: int a=2, b=0
if (a=b)
cout << "AA";
Cout << a << "\t" << b;

0 0

a=0, b=0 في المثال السابق

AA0 0

* ملاحظة مهمة: إذا لم يوجد if فإن الكود الذي بعدها يعتبر syntax error ← هو ليس الكود الذي لدى main تعتبر. if يجب أن تكون موجودة

المحاضرة 10

```
Ex: int x;
    cout<<"Enter integer value: ";
    cin>>x;
    if(0<=x && x<=10)
    cout<<x<<" is in [0,10]\n";
```

في هذا البرنامج أي رقم بطبعه (User)، ويكون بين [0,10] راح يظهر له رسالة بالواجهة، غير هيك مش راح يطبع أشي.

// Check whether a given value is in the range [0,10] "

```
Ex: int x;
    cin>>x;
    if(0<=x<=10)
    cout<<"yes";
```

* لاحظ في هذا المثال: if(0<=x<=10) ⟸ الأولوية تبدأ من اليسار، لأنه لهلة متساوية، لو أنا دخلت قيمة لـ x أكبر من 0، راح يطبعلي true وهذا

1 <= 10 ⟸ true <= 10 : إذا هيك راح يصير بشكل

1 ⟸ true ⟸ true وجوابها ⟸ 1 <= 10

فبالتالي راح يطبع true، طب لو دخلت x قيمة أقل من صفر

0 <= 10 ⟸ 0 وهذا false ⟸ false <= 10 بنصير هيك

↳ وهاي جوابها true، فبالتالي راح يطبع Yes.

Scanned with CamScanner

* لاتنسى إنه جملة If ما بتنفذ إلا إذا كانت true.
* لاتنسى إنه نهاية كل عملية مقارنة دايمًا بـ true أو false.

* في المثال اللي فات "Yes" في كل الأحوال طبعًا، في كل الأحوال وكل الأرقام اللي دخلتها كانت true، وهاي لهالدرجة بيّن حتى منيح إنه كل مرة يعطيني نفس النتيجة فا بنتاهيك.

* بدل "AND &&" اقدر أتخلص منها باستخدام "||"

Ex: Write a programme in C++, to print Any Capital letter in English language that Input by the User ?

Sol:

```
#include <iostream>
Using namespace std;
Void main ( )
{ Char x;
Cout<<"Enter capital letter:\n";
Cin >> x;
If (x >= 'A' && x <= 'Z')
Cout<<"The capital letter is:"<<x; }
```

Ex:

```
int x;
Cin>>x;
if(x>0) [;] <-
Cout<<"positive";
```

positive

هذا المثال شو عم يدخل User، عم يطبع كلمة Positive

لأنه وجود الـ [;] بعد جملة If، أنهت وجودها كملكي عم تلغي أثرها.

Scanned with CamScanner

* المثال السابق يُعتبر "Logical error" إذا لم تضع if

* المثال السابق سوف يرفض User من إدخال أي رقم غير positive.

* Two_way selection: "If-else":

* Syntax:
```
if(expression)
    statement;
else
    statement;
```

* ملاحظات مهمة:

1. إذا وضعت فاصلة منقوطة بعد if الأولى ⇐ syntax error
لأن else ستبطل تابعة لها.

2. إذا وضعت بعد else ، الجملة التي تليها تصبح تابعة لـ main.

* لازم تتذكر أنه إذا بدي أحد جملة شرط لازم أستقل بـ braces، واستخدام braces في جملة شرط الواحدة هي اختياري "زي ما بدك".

User

Ex: int x;
Cout << "Enter integer value:\n";
Cin >> x;
if (x % 2 == 0)
Cout << x << ": Even\n";
else
Cout << x << ": odd\n";

x = 5
5: odd

User
x = 12
12: even

* تذكر: إذا وضعت ; نهاية جملة if ينتج syntax error

* وإذا وضعت ; بعد else لم نعد نحتاج لوجودها والكود الآتي تحت نعمل لا main.

Ex: int x = 0;
if (x++)
{ x += 2; }
Cout << x;
else
x--;
Cout << "\n x = " << x;

x = 0

* وإذا وضعت braces ← syntax error لأن else لم تعد تتفصل عن if.

* لا else لازم تكون تابعة لـ If.

Scanned with CamScanner

* في المثال السابق: استخدم القيمة الابتدائية لـ x الى هي 0، ولما ها كتدل لقيمة تبعير [1]، وعندك كتب x-- تبعير 0.

If-else حالة

* تذكر دائما اذا كان من جملة جوة الـ If، اجباري الـ braces

* في المثال السابق: لو بدل x--، كانت x-=3

الـ output الناتج | x=-2

Ex: int x=5;
if(x++)
{
x+=2;
cout<<x;
else
x--;
cout<<"\nx="<<x;

output:
8
x=8

* Conditional operator (? :)

* زي ما تكلمنا هو عبارة عن بديل لجملة الـ "else"، لكن شرط استخدام هذا الـ operator هو انه يكون كتلة جملة وحدة بس، فازبط اكتر من جملة.

* syntax:

(expression1) ? (expression2) : (expression3);
بـ true الـ 1 | false الـ 0

Ex: Rewrite the following expression:

```
if (x%2)==0
Cout<<"Even";
  else
Cout<<"odd";
```

Sol:

1 ← `(x%2==0) ? Cout<<"Even" : Cout<<"odd";`

2 ← `Cout<<(x%2==0 ? "Even" : "odd");`

Ex:

```
int x=5, y=7, z;
z=x>y ? x : y;
Cout<<z;
```

7

هذين هاي البلوك ← `z=x>y ? x : y;`

تقدر تكتبها هيك ← `(x>y ? z=x : z=y);`

هذين البلوك اللي فوق ممكن نفرها هاي ↓

```
If(x>y)
z=x;
  else
z=y;
Cout<<z;
```

Scanned with CamScanner

⊛ Nested If:

Ex:
```
#include <iostream>
using namespace std;
int main( )
{ int x=6 ; int y=2;
if (x>y)
cout <<"x is greater than y\n";
else if (y>x)
cout <<"y is greater than x\n";
else
cout <<"x and y are equal \n";
```

```
x is greater than y.
```

⊛ في المثال السابق لو $x=2, y=6$

```
y is greater than x.
```

⊛ المثال السابق لو $x=2, y=2$

```
x and y are equal
```

⊛ افرض انني بدي أدخل علامات الطلاب بحيث إلي طالب

[90, 100] → A [60, 69] → D

[80, 89] → B [50, 59] → E

[70, 79] → C < 50 → F

Scanned with CamScanner

* هل ثمة وظيفة أخرى لـ If وهي إيقاف البرنامج كبير أو كبير جداً، الكود التالي.

Sol:
```
int m;
cin >> m;
if (m >= 90 && m <= 100)
cout << "A";
else if (m >= 80)
cout << 'B';
else
if (m >= 70)
cout << "C";
else if (m >= 60)
cout << "D";
else if (m >= 50)
cout << "E";
```

أي ليوقف البرنامج عند الجملة التي يتحقق عندها الشرط.

لصيد كل الشروط ولصيد لا.

* كل حالة وجه هاهنا الحل من ثمة وظيفة لقدام ضلات هل الهاي القضية.

* خفت ليش اشتغلوا كل الـ If Nested كان ما يفهم كل شوي يكتبوا الحالة If, If هاي تشغل وقت كثير.

* لـ switch لم نحن وأمين لأنه Cases ما يحبها.

* الـ Semi colon لـ If ما بيخلي والو.

Scanned with CamScanner

⊙ هذا برنامج بيقول للمستخدم يدخل حرف بالإنجليزي ويطلع له إذا كان Capital ولا small ولا ....

Ex: Char ch;
cin >> ch;
if(ch >= 'A' && ch <= 'Z')
cout << ch << " is Capital \n";
else if(ch >= 'a' && ch <= 'z')
cout << ch << " is small \n";
else
cout << "In-valid Alphapet letter";

* المحاضرة رقم (١٢):

⊛ Nested if:

User x=11, y=0
~~** out~~

x=-11, y=2
** out

int x, y;
cin >> x >> y;
if(x && y)
{if(x>5)
cout << "**";
else
cout << "**" << "out";}

⊛ الفرق بين الـ If الأولى والثانية هو إنه الـ else بترجع لأول If قبلها، طول فيه مجال إنه فيه true بالـ If الأولى، وطول فيه Braces بيعطي تابع الـ If الأولى.

⊛ إذا تابع الـ If الأولى false فقط بتطبع out

false ولا، ما بنفذ true ولا، تلقائي ...
else لأن

Ex: int x, y;
cin >> x >> y;
if (x && y)
{ if (x>5)
cout << "x x";
}
else
cout << "* *"; cout << "out";

x=1, y=0
** out

x=11, y=2
** out

x=11, y=2
out

* ملاحظة: ما بحتاج لأقول else لأن طول ما الكوديشن بتيجي true ← AND لـ
* إذا بالأقواس (braces) ظلت على آخر if لـ else لمين بتبع؟

* switch: multiple selection structure.

* switch syntax:

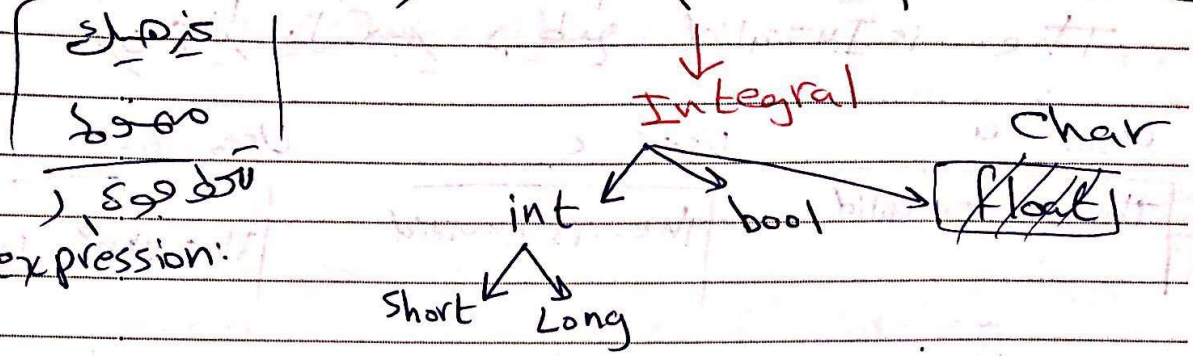
switch (expression) → متغير
{ case value 1: st1; st2; ... break;
case value 2: st1; ... break;
default: ... وجودها اختياري
}

* Constant integer يكون ← value لـ case
values.

Ex: case 1: ; case True: ←

Scanned with CamScanner

* ملاحظة: في الـ expression لازم يكون:



* لو الـ expression جواه string أو double يعتبر syntax error.

* الـ value اللي بتتحط في الـ case لازم تكون صحيحة integer يعني 1, 2, true, 'A' وهكذا.

* braces of the statement is not needed.

Slide 58:

```
Char grade;
Cin>>grade;
switch (grade)
{ Case 'A': Cout<<"The grade is 4.0";
  break;
  Case 'B': Cout<<"The grade is 3.0"; break;
  Case 'C': Cout<<"The grade is 2.0"; break;
  Case 'D': Cout<<"The grade is 1.0"; break;
  Case 'F': Cout<<"The grade is 0.0"; break;
  default:
  Cout<<"The grade is invalid";
}
```

Scanned with CamScanner

* في المثال السابق: لو دخل المستخدم الرموز الـ Small. فإن البرنامج سيطبع The -- is Invalid. لذلك يجب أن نضيف الرموز الصغيرة بدل كبيرة.

User: a
The -- is Invalid

User: c
The -- is Invalid

User: 65
The grade is Invalid

* طيب ممكن بدل ما أكتب الحالات بهذي الطريقة أو أستخدم OR داخل الـ switch كيف؟

```
Char grade;
Cin >> grade;
Switch (grade)
{ Case 'A':
Case 'a': Cout << "The grade is 4.0";
   break;
Case 'b':
Case 'B': Cout << "The grade is 3"; break;
   .
   .
   .
default:
Cout << "The grade is Invalid";
}
```

* فائدة البرنامج لغير مبتدأ OR، لغير أكتب الـ Cases بشكل بعضها، ولغير مكتب بعض.

Scanned with CamScanner

Ex: Slide 59:

لو دخلت الرقم 1 بيطلعلي F

```
int score, char grade;
cin >> score;
switch (score/10)
{
 Case 0:
 Case 1:
 Case 2: Case 3:
 Case 4: Case 5: grade = 'F'; break;
 Case 6: grade = 'D'; break;
 Case 7: grade = 'C'; break;
 Case 8: grade = 'B'; break;
 Case 9:
 Case 10: grade = 'A'; break;
 default:
 cout << "Invalid test score"; }
```

User: 75 → C

User: 44 → F

User: 110 → Invalid

⊛ ملاحظات وهمة:

١) إنما متفقين إنه الـ switch مصنوع يدخل له أرقام من نوع Integer، لمين فلاز لو الـ User دخل له أرقام من 61 إلى 69 إع يتقبلها الـ switch إنه 60 فقط وللي فوق ويتعتبر الـ Case رقم 6 طب ليش؟ لأنه إنا أعلاز كامل إنا الـ Declaration لـ score إنه الرقم بدي إياه Integer، فإنا بالنسبة للـ switch إذا دخلت قيمة كليات صحيحة لأرقام من نوع Integer، طلع بنتائج كس، تلقائيًا الـ switch إع يأخذ الجزء الصحيح، ويهمل الجزء العشري، ولهذا يوظف الـ Case قيمة

٢) إنما قيمة الـ score كل ١٠، حتى وأنا كل 100 Case، وبتشغل طالما إنه الـ switch يهمل الجزء العشري، بهاي الطريقة لقسم أنا كل كل ١٠ الـ Case / Cases وحدة فقط.

⊛ لو بدك تتعرف الـ score كمان إنه float بتعمل الـ Cast قبل الـ switch.

Scanned with CamScanner

المحاضرة رقم "13"

Ex: Find syntax errors in this program.

```
1 float sum = 0 , x ;
2 cin >> x ;
3 switch ((x++)
4 { case 1:
5 Case 4 ; sum += x ; x++ ; break ;
6 Case2 : sum -= 10 ; break ;
7 Case true : x++ ; break ;
8 Case x+10 : --x ; break ;
9 Case 2+2 : x+=4 ; sum += x ;
10 default : cout << "Finish" ; }
```

Integral هي switch ، وتأتي بعدها قوسين ()
وإذا بالقوس الأول معرف x ، float ، كمتغير صحيح.

switch (static_cast<int>(x++))

ج) القوس الخاصة لا فوقها ، و Case 4 ، هاي ما بتنفع
لازم تكون بدايتها :

Uselesscase هاي "Case2" ، هاي موجودة :
Unused case

لازم تكتب Case 2

Logical error

وليس syntax.

Scanned with CamScanner

(1) هوك / expression (2) هوك / cases (3) فو : (4) Duplicated (5) فإن :

(4) لاحظ البكج / ابة و البكج / ابة بعضهم Duplicated Case

Case 1 يعني هة فعليا أنا في عندي بكيف بأيم

لأنه / true هو بات قيمة (1) البكج / ابة Case (1) أصلاً فيها وامر فو "دبل كيس".

(5) نفس الإشي بالنسبة للبكل / رقم "9" و "5" Duplicated, Case.

(6) البكج / رقم [8] ⇐ لاحظ ماقط هوك / Case ← (x+10) هاي / كلام ما بصير لأنه x متغير ممنوع تحط هوك / Case إلا ثوابت.

(*) لا تنسى إنه / Duplicated Case هة فعليا Syntax error.

Ex: Case 'A':
Case 65: ⇒ Duplicate هاي / الحالة فعليا

(*) تذكر : هوك / expression لازم ثوابت (Integral) تبع / switch.

(*) هوك / Cases / Int, Const, Char, true, false

'A', true, 66

(*) يكون فيه space بعد / Case.

(*) إنه ما يكون فيه Duplicated.

Scanned with CamScanner

⊗ تقدر تخلي كل الـ cases اللي عندك، وتشملهم بحالة خاطئة واحدة فقط بالنهاية بعد الـ default.

Ex:
```
float sum=0;
int x; cin>>x;
switch(x++)
{ case 0:
  case 1: x+=3; sum+=x;
  case 2: x++; sum+=x; break;
  case 5: x=5; break;
}
cout<<x<<"\t"<<sum;
```

واحنا دايماً break هو موجود ولا لا.

[1] User: x=1, sum=0

6	11

[2] User: x=0, sum=0

5	9

[3] User: x=10, sum=0

Case ⊗ ثم

11	0

++10

[4] User: x=5, sum=0

5	0

5 بتطبع 6 بتطبع

write a simple C++ Calculater using switch.

Ex:

```
please Enter two numbers
10   20
please Enter the operation
  +
10+20 =30
```

```
float n1, n2;
char op;
Cout<<"Enter two numbers:";
cin>>n1>>n2;
Cout<<"Enter the operation:";
Cin>>op;
switch (op)
{ Case '+':
  cout<<n1<<op<<n2<<"="<<n1+n2<<endl;
  break;
  Case '-':
  Cout<<n1<<op<<n2<<"="<<n1-n2<<endl;
  break;
  Case '%':

Cout<<n1<<op<<n2<<"="<<static_cast<int>(n1)%
static_cast<int>(n2)<<endl; break;

      default:
Cout<<"Invalid operation";
```

Scanned with CamScanner

Chapter "5": Control structure II (Loops):

1. for
2. while
3. do while.

1) for:

Syntax:

```
for (initialize control var ; Loop continuation test ; Update control var)
    { st1 ;
      st2 ;
    }
```

* وجود كلمة واحدة او اكثر من كلمة لازم braces { } الـ

Ex: Write a program to print integers for 1 to 10.

```
int ctr ;
for ( ctr=1 ; ctr<=10 ; ctr++ )
    { cout << ctr << "\t" ; }
cout << endl ;
```

(1) (2) (3) (4)

for الـ يمكن عدم كتابة أي جزء من أجزاء الـ for و pre او post

الـ for لا يوجد بعدها Semi Colon (الفاصلة) لأنها تعتبر جزء من for

* من الأخطاء الشائعة عند كتابة for عدم كتابة ; بين أجزاء الـ for (Logical error.)

* لو حاضنة braces بعد for
update variable
لصح تابعة لـ main وبس
لـ for، هاي بتطلع بسوها
Infinite Loop. وبتفضل
للصح وامر.

Ex: int ctr = 1;
for(; ctr <= 10 ;)
{ cout << ctr << "\t";
ctr ++ ; }
cout << endl;

* مع وجود braces، بتطبع الأرقام من 1 لـ 10.

* ملاحظة عن for: Initialize Control variable ← وجوده بس لازم يكون موجود

* ملاحظة: لو حطيت ; بعد for حتشتغل for (infinite Loop).

Ex: for(int ctr=1 ;) ← بدل أكتبها هيك
من Declaration
وبكون بس جوة for.

Ex: int ctr ;
for(ctr = 1 ; ++ctr <= 10 ; ctr ++)
cout << ctr ;
cout << "ctr = " << ctr ;

تذكر كمان فيه أولوية
جوة قلب for،
بتطلع false ← بتطلع 5
لـ Loop كلها وبتطلع
القيمة.

2 4 6 8 10
ctr = 12

Scanned with CamScanner

Ex: int ctr;
for(ctr=1; ctr<=5;)
cout<<++ctr;

2 3 4 5 6

Ex: int ctr;
for(ctr=1; ctr<=5;)
cout<<ctr++;

1 2 3 4 5

ملاحظة: لو كانت ctr++ مش جوا الـ for هتطبع 6 لأن الـ expression <<++ctr

Ex: print all numbers that are divisable by 3.
from 1 → 100 and count them.

Sol:
```
int c=0 , int c3=0 ;
for(c=1; c<=100; ++c)
{ If(c%3==0)
{cout<<c<<"\t";
c3++; }
}
cout<<"The number="<<c3;
```

بيطلع لي الـ for أول ما يقف رقم أكبر من 100.

لو وقفت في الـ for بعد الـ braces الـ If هينفذ c3++ بس.

لو وقفت في الـ for جوا الـ If braces هيطلع 0

3 6 9 12 .. 1683

sum of them

Scanned with CamScanner

* while:

```
// initialize control variable
while (Loop continuation test)
{   st
    :
    Update control variable ;
    :
}
```

(1) print integers 1-10 Using while:

Sol:
```
int ctr=1;
while ( ctr<=10)
{ cout<<ctr<<"\t";
  ctr++; }
```

* لو حطيت braces (Infinite Loop)
* لازم أحط braces كمان أضيف الـ ctr++ تابعة لـ Loop.

Counter ; متغير التحكم Controlled Loop

Ex:
```
int ctr=1;
while(ctr++<=10)
cout<<ctr;
```

2 3 4 5 6 7 8 9 10 11

Ex:
```
int ctr=1;
while(++ctr<=10)
cout<<ctr;
```

2 3 4 5 6 7 8 9 10

Scanned with CamScanner

المحاضرة رقم "15"

✳ While Loop:

1. Counter controlled Loop.
2. Sentinal // //.
3. flag // //.

✳ مثال على الحالة الأولى.

Ex: Find the summation of odd numbers [1, 10] ?

```
int ctr = 1 ;
int sum = 0 ;
while (ctr <= 10)
{
    sum += ctr ;
    ctr += 2 ;
}
cout << "sum of odd numbers" << sum ;
```

✳ أكثر مشكلة لا while loop هو Infinite تتكون بسبب الـ braces { } Loop.

✳ الـ Loop ← 11 مرة، بينما الـ ctr ← 25 مرة.

✳ من المثال السابق لو كانت جملة الطباعة خارج الـ braces { } فإنها تطبع أول باقي رقم.

✳ ملاحظة: أي متغير يطلب ناتجه لازم يتعمل Intialize بـ zero، وذلك لتجنب الـ Rubbish وتجنب (Logical error).

✳ طريقة مختلفة للمثال السابق:

```
int ctr ;
int sum = 0 ;
for (ctr = 1 ; ctr <= 10 ; ctr += 2)
    sum += ctr ;
cout << sum ;
```

Scanned with CamScanner

* لفتن بالمثال السابق لو طلب حزب " product " حيث بلقى:

```
int ctr=1 ;
int prod=1 ;
while (ctr<=10)
{ prod*=ctr;
  ctr+=2; }
cout<<"product of odd numbers"<<prod;
```

* في المثال السابق أول قسمع حاصل ضرب بـ/ بالك تقل Intialize لـ prod بـ صفر، لأنه هيا تضرب في أول عدد صفر.

[2] "Type 2" : sentinal : controlled Loop :
هي القيمة اللي تؤدي عن إيقاف الدوران، بحيث كلما الدوران تكمل كلها، وكلما الدوران تقف عند كلمة الإدخال.

Syntax:

```
[1]<--- cin >> variable ;
while (variable != sentinal variable)
{
[2] <--- cin >> variable ;
}
```

Ex:

```
int sum=0 , v;
--->cin >> v;
  while (v != -1)
  { sum+=v;
--->cin >> v; }
  cout<<sum;
  cout<<v;
```

sentinal value : هي القيمة اللي بتوقف الـ Loop.

User: v: 5, 7, -3, 1, -1

10	-1

ظهم كلهم شمال

Scanned with CamScanner

* ملاحظة مهمة جدا: في المثال أو الأسئلة السابقة تذكر فيجب إنه الـ Loop بطلعلي بـ 16 يومن ينتج عنه Loop كمل الـ Loop "false".

Infinite Loop. ، الثانية cin الحالة

* في المثال السابق لو بدلت الترتيب:

```
int sum=0;
int v;
cin>>v;
while(v!=-1)
{ cin>>v;
  sum+=v; }
cout<<sum;
cout<<v;
```

User: 2,3,4,5,-1

11 -1

* في المثال السابق لو كانا طالب في فكرة الأرقام اللي راح يتم إدخالها ماعدا الأرقام السالبة، يعني إذا دخلت رقم سالب توقف البرنامج.

Sol.

User: 1,4,5,7,-1

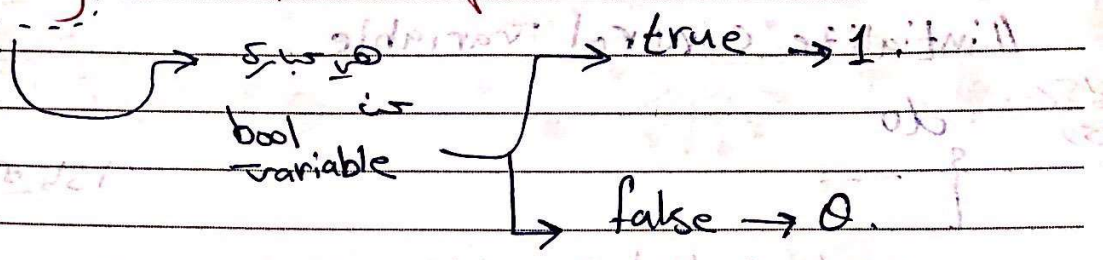
```
int sum=0, v;
cin>>v;
while(v>0)
{ sum+=v;
  cin>>v; }
cout<<sum;
cout<<v;
```

17 -1

* وظيفة cin جوة الـ Loop ← while/sentinal

* تذكر فحص شيء بطلعلك Loop لازم يكون صواب خلص.

[3] Flag Controlled Loop:



Ex: bool flag = 0; int ctr = 1;
while(! flag)
{ cout << ctr;
If (ctr % 5) == 0)
flag = true;
ctr++;
}

1 2 3 4 5

* عند رقم (5) يصبح flag يساوي true فيقف الـ Loop.

* المتغير المتبقي القيمة المخزنة هو الـ Counter ← 6.

Ex: bool flag = true;
int ctr = 1;
while (flag)
{ cout << ctr;
If(ctr % 5 == 0)
flag = false;
ctr++;
}

نفس المثال السابق لصيغة مختلفة.

flag Controlled Loop.

* prime numbers: 1, 3, 5, 7, 11, 13, 17.

Scanned with CamScanner

⊛ Do-while:

```
//intialize control variable
do
{
   ...
   update control variable
} while (Loop Continuation test);
```

لازم تحط ; بعد while وإلا syntax error.

Ex: print integers 1-10:

sol:

```
int c=1;
do {
cout<<c;
c++;
} while (c<=10);
```

output: 1 ... 10

⊛ مثال: لو c=15 بدل c=1، كم رح يطبع؟

c=15 16

output: 15

⊛ ملاحظة: جملة do، كم مرة على الأقل رح تنفذ؟ مرة واحدة حتى لو ما تحقق شرط while.

Scanned with CamScanner

⊛ لا مطرق لـ while، تصير الكل اللي بتكتبها تابعة لـ main يعني لازم ع الـ Loop كامل، لو تتفق لا يطلع برة الـ Loop (بفك الأقواس braces)

Ex: int x=5;
while(x<10)
{ cout<<x;
x++; }

5 6 7 8 9

⊛ لو طيحت ; بعد while، يعني Infinite Loop، وبقف أطبع: يرقش بالشاشة.

⊛ ولو بدي أخلص من مشكلة Infinite Loop، هو موجود لو بكتب x++ بالـ while ← الحل: رجع لـ 16.

⊛ Random numbers: (أرقام عشوائية)
→ rand( ) ← generate random numbers [0-32767].

⊛ كمان لازم أعرف وظيفتها: #include<cstdlib>

shifting value [a, b] ← a بدي أبلش منها
بحيث أخلي الـ zero بالطرف

scaling factor.

⊛ طريقة استخدام ← a+rand( )%(b-a+1)

⊛ الهدف منها أو الفكرة منها ← generate Random numbers → (بالفترة المعينة)

Ex: [1,6] ⇒ ~~1+rand%~~ 1+rand( )%(6-1+1)
1+rand( )%6 → لازم هاي الصيغة داخلة بالبرنامج أعرف منها إنو cout أو int.

Scanned with CamScanner

Ex: Write a C++ program, to generate 100 random numbers [3,15].

Sol:

```
#include<iostream>
#include<cstdlib>
using namespace std;
void main( ) {
for(int i=0; i<100; i++)
cout<<(3+rand( ) %13)<<endl;
}
```

* ملاحظة :- طبعا 100 فوق البرنامج ، لأنه أخذت بمتغير لكتير ، one error ← يا بقل 99 في وراك ، بأخذ أقل من 100.

Ex: [-3,15] ⟶ -3+rand( ) %19

⟶ rand( ) %19-3

* وممكن يطلع البرنامج وبالآلة من الجزء :

rand( ) %10-1 ⟶ a=-1

b-a+1=10 ⇒ b+2=10 ⇒ b=8

[-1, 8]

* Using break and Continue with Loops:

```
Ex: int i=5;
while(i<20)
{ if(i%2)
    break;
  i++;
}
cout<<"i="<<i;
```

→ cout<<i<<endl;

5
i=5

Scanned with CamScanner

* ملاحظة كل برنامج بالبرنامج يوصل لكلمة break بيطلع برة الـ Loop كامل.

Ex: int i = 5;
```
while (i < 20)
{ cout << i << endl;
  if (i % 2)
  continue;
  i++;
}
cout << "i = " << i;
```

5
5
⋮
Infinite Loop

* في المثال اللي فات كلمة continue (الاستمرار) كلمت skip عن الجملة اللي تحتها وكملت الـ Loop.

* دائمًا بال while و do while، لما كانت كلمة Update قبل كلمة continue، تكون النتيجة Infinite Loop.

Ex: int i = 5;
```
while (++i < 20)
{ if(i % 2)
  continue;
  cout << i << endl;
}
cout << "i = " << i;
```

6
8
10
12
14
16
18
i = 20

Scanned with CamScanner

* مثال سابق لو بدي أطبع الأرقام من 1 إلى 20

```
int i=5;
while(++i<20)
{ If(i%2==0)
continue;
cout<<i<<endl;
}
cout<<"i="<<i;
```

Ex:
```
for(int i=1; i<=15; i++)
{ if(i==5)
  break;
  cout<<i;
}
```

i: 5

1 2 3 4

لما القيمة اللي طلعت 5 الـ Loop بتوقف.

Ex:
```
for(int i=1; i<=15; i++)
{ if(i==5)
  Continue;
  cout<<i;
}
```

i: 16

1 2 4 6 7 8 ... 15

* طول ما الـ for فيها Update Controlled variable ← موجود

Continue في for: مش بتخلي الـ Loop Infinite

* مثال سابق لو حطيت بدل i==5 ، i%5 ، طبع

5 10 15

Scanned with CamScanner

```
Ex: int i=1;
for (i=1; i<10; i++)
{ if (i==3 || i==6)
  continue;
  cout<<i;
  i++;
}
cout<<i;
```

```
1 4 7 9 11
```

← continue لو وقفت بدل break، بيروح لهنا.

⊛ Slide (45+46) → Ch.5:

```
Ex: int sum=0, num;
    cin>>num;
    while (cin)
{ If (num<0)
  { cout<<"-ve number found\n";
    break; }
    sum=sum+num;
    cin>>num;
}
    cout<<sum;
```

Sentinal controlled Loop

Loop الحلقة اللي كلمتين، ك، ل

Case1: 10, 20, 30, -4, 5, 6

```
-ve number found
60
```

⊛ من المثال اللي فات، الـ sentinal هون موجبة عكس الـ Int، فإذا أدخلت حرف من الـ Input failiur

Case 2 : 10, 20, 30, A

60

char

Input فقط كتابة
failiur

⊛ Slide 46:-

```
int sum =0 ,num;
Cin>> num;
while( Cin)
{ if (.hum<0)
{ Cout<<"- ive number found \n";
  Cin>>num;
  Continue; }
  sum= sum+num;
  Cin>> hum;
  }
  Cout<<sum;
```

⊛ Nested Loops:

```
int H; Cin>>H; int i, j;
for( i=1; i<=H; i++)
  { for(j=1; j<=i; j++)
    Cout<<"*";
    Cout<<endl;
    }
```

*
**
***

⊛ ملاحظة: اللوب الأولانى عشان عدد الصفوف، واللوب Loop
الثانية عشان محتويات الصفوف.

Scanned with CamScanner

⊛ من المثال السابق لو بدي أكتبها بالعكس:

```
for(i=4; i>=1; i--)
```

⊛ Multiplication table:

```
1 2 3 ---- 10
2 4 6 ---- 20
3 6 9 --- 30
4 8 12 --- 40
5 10 15 -- 50
```

Sol:

```
int r, c;
for(r=1; r<=5; r++)
{ for(c=1; c<=10; c++)
  cout<<r*c<<"\t";
  cout<<endl;
}
```

(حكاوز 18)

Ex: nested Loop اكتب هذا البرنامج بال

```
1 * * * *
1 2 * * *
1 2 3 * *
1 2 3 4 5
```

Sol:

```
int i, j, k;
for(i=1; i<=5; i++)
{ for(j=1; j<=i; j++)
  cout<<j;
for(k<=5-i; k>=1; k--)
cout<<"*";
cout<<endl;
}
```

Scanned with CamScanner

كيف أفاهم
كلهم كلام ترتيب ؟

Ex write a c++ program that promet the User to input an integer and then out put it as individual digits and find the sum of these digits.

```
sol.. int n,d,temp;
      int sum=0;
      Cout<<"integer number:";
      cin>>n;
      temp=n;
      while(temp!=0)
      {d=temp%10;
       sum+=d;
       temp/=10;
       Cout<<d<<"\t";}
      Cout<<endl<<"sum="<<sum;
```

---

1. الأعداد الزوجية وعكوسها.
2. الأعداد الزوجية ومجموعها.
3. الأعداد التي تبدأ بقيمة ولا تنتهي بنفس القيمة وعكوسها.
4. أكتب رقم وقل بالعربي ولا تزد عن ثلاثة.
5. أكتب رقم وقل كم منزلة فيه.
6. Decimal → Binary / Decimal ← Binary

Scanned with CamScanner

Ex:

1	2	3	4	5
1	3	5	7	
1	4	7		
1	5			
1				

Sol:

```
int i,j;
for(i=1; i<6; i++)
{ for(j=1; j<8; j+=i)
  { If(j==6) break;
    cout<<j; }
  cout<<endl; }
```

1. انتبه على الـ Loop، كم مرة وكم عمود بدي.
2. الـ Loop الخارجي يحدد الصفوف، والـ Loop الداخلي يحدد الأعمدة.
3. انتبه الـ Increment، هل زيادة ثابتة أم متغيرة؟ "فلتة كلمة تعرف منها زيادة".
4. لمرين هتنبه كل اللي فوق باش مي تقم وحشية.

1. دفتر (Loops).
2. ورق طارحة (Loops).
3. دور على ~~الحبر~~ (Loops).

Scanned with CamScanner

" لا يوجد شيء "

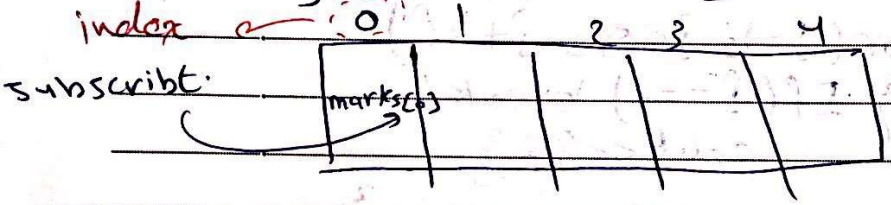
Chapter 8: Arrays and strings.

* Array: a Collection of a fixed numbers of Component, all of the same type.

- Value of index is the position of the item or (element) in the array.

. Array في العنصر هو الذي يحدد index القيمة

- Array index starts at zero. أي أن



- [ ] ⇒ subscripting operator.

* Array Declaration:

* syntax:

Type Array Name [size]; ← # of items.

Ex: ① double marks [5];
② int Array [7];
③ int A1[5], A2[7]; Array من نوع 7 عناصر marks
④ char c, ch[5];

subscript operator: لإلى العنصر في Array نستخدم *

(*) ملاحظة: معاملة العنصر الواحد من مصفوفة Array كأنه متغير variable عادي

⇒ Array Intialization:

→ ① Intialize list { , , , }

→ ② Loops (for)

① Intialize list:

Ex: int x1[4] = {5, 6, -3, 4};

	0	1	2	3
x1	5	6	-3	4

↳ Constant pointer.

index رقم العنصر

x1[2] = 10; ← ② كيف لو أردت تغيير قيمته:

② رقم index العنصر الذي موجود بحيث over right مكانة. وتريد تغيير قيمته ب 10

(Access on third element)

(*)

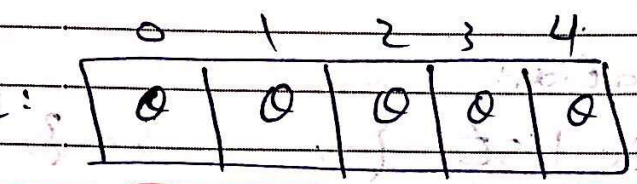
First element ⇒ x1[0] = 5 ✓

(*) x1[3] = x1[0] + x1[1];

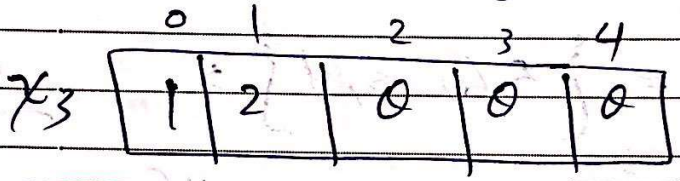
11 = 5 + 6 ← index رقم العنصر الذي موجود ②

⑪ تخزن في 4 ← over right

Ex: int x2[5] = {0}; initialize x2 with zeros



Ex: int x3[5] = {1,2};



* Ex: int x4[5] = { }; syntax error

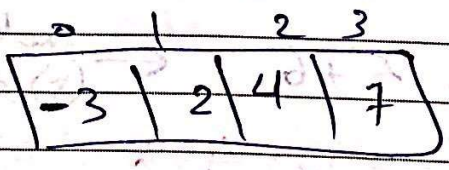
Ex: int x5[5] = {1,2,4,6,8,7}

syntax error ← Compailer

5 is size, 6 is number of elements

Ex: int x6[ ] = {-3,2,4,7};

Constant integer size.



كود لما مبدتش size هو

هو بوفره من element

Ex: int z [ ] = {false, 'B', 'd', 1.5};

implicit Casting من شان الـ Compiler.

0	66	100	1

* note: Array داخل elements معاملة زي ما بنعامل loop variables.

* Intialize arrays by Loops:

```
int A[20], i;
for(i=0; i<20; i++)
  cin>> A[i];
```

intialize Arrays from User input.

* لما بدي أعمل loop بتعامل مع Array بدي أخليها من i=0 لحد size-1.

* Intialize array with Random numbers [1, 10]

Hint: #include <cstdlib>

```
int A[20], i;
for(i=0; i<20; i++)
A[i] = 1+rand( )% 10;
```

* لو قلنا عددتين و rand( )

Hint: #include <cmath>
int Arr[5];
for(int i=0; i<5; i++)
Arr[i] = pow(2, i);

	0	1	2	3	4
Arr	1	2	4	8	16

// Cout << Arr; → Address for Array
// Cin >> Arr; → syntax error

// print Array elements
for(int j=0; j<5; j++)
cout << Arr[j] << "\t";
cout << endl;

Ex: float f[4], i;
for(i=1; i<=4; i++)
f[i] += 5;

- errors Logical

Rabbidy
1) Initialize كامل Array وباستخدام for loop
2) كامل Skip ومش مطلوب حد يدخل Array من عنده.
3) خلي بالك دايماً من size، مش لازم تتخطى حدود الـ Array "out of Boundary Array"

⊛ int x[5]; Syntax error
x[5]={5,4,3,2,1};

⊛ Declare and intialize in one statement
"20 5 9 56"

⊛ Array size:
- must be constant integer ↔ value / variable.

Ex: int x[10]; ←
int size=7;
float a[size]; ∝ syntax error.

Const int size=7;
float a[size]; ←

Ex: find the sum and average of array {2, 4, 7, 3, 5}.

Sol: const int size=5;
int A[size]={'A',7,3,5};
int sum=0,i;
float avg=0;
for(i=0;i<size;i++)
{ sum+=A[i]; }
cout<<"sum="<<sum<<endl;
avg=sum/static_cast<float>(size);
cout<<"avg="<<avg;

Scanned with CamScanner

Ex: find the Largest element in Array :

sol:
```
const int size=5;
float f[size]={1.5, -3.2, 4, 7.2, -1.5};
    float Largest = f[0];
    for (int i=1; i<size; i++)
    { if (f[i]>Largest)
        Largest = f[i]; }
    Cout<<"Largest element is:"<<Largest;
```

(*) search for value in Array:

```
int x[5] = {4,3,7,-2,1};
int value;
Cout <<"inter integer value";
Cin>>value;
for(int i=0; i<5; i++)
{ If(x[i]==value)
    { cout<<value<<" found\n";
        break; }
}

    If (i==5)
      cout<<value<<" Not found\n";
```

Scanned with CamScanner

**

Ex: int values[10] = {5, 7, 3, 2, 4, 2, 1, 8, 9, 10};
int odd[10], even[10];
int odd_index = 0, even_index = 0;
for(int i = 0; i < 10; i++)
{ If(values[i] % 2 == 1)
{ odd[odd_index] = values[i]; odd_index++; }
else
{ even[even_index] = values[i];
even_index++; }
}

"21 Slide"

⊛ Aggregate operation: واني ماقدر أتعامل مع (Array) بشكل كامل
فقط كصفر صفر.

int list[5] = {0, 4, 8, 12, 16};
int list2[5];

// Copy list1 to list2;

list2 = list1; ✗ // illegal ⇒ Syntax error.

⊛ Note: Array name is a constant pointer.
وأشير كمتغير وكانه "بداية" بس نوعه فقط.

Scanned with CamScanner

Ex: for (int i=0; i<5; i++)
{ List2[i] = List1[i]; }
cin >> List2;
cout << list1;

print the address of first element in List1 ⇒ Array

* Parallel Array:

Ex: int stdID[10];
Char Grade[10];
for(int i=0; i<10; i++)
{ cout << "enter stud id";
cin >> stdID[i];
cout << "enter stud grade";
cin >> Grade[i]; }

for (int j=0; j<10; j++)
cout << stdID[j] << "Grade is:" <<
Grade[j] << endl;

* Base Address of an Array:

- Array name: is a constant pointer that contains the address of the first Array element.

Ex: char n[2][5] = {'a', 'b', 'c', 'd', 'e', "Amer"};
for(int i=0; i<2; i++)
{ for(int j=0; j<5; j++)
cout << n[i][j];

allocate Amer

4Byte كل تحجز
مع بعض.

Ex: int mylist[5];
(// base address = 1000)
عنوان أول عنصر موجود في Array.

0	1000
1	1004
2	1008
3	1012
4	1016

Ex// double تأخذ كل 8 مع بعض
bool تأخذ 1 كل مع بعض

⊛ Two and multi dimensional Array:

x(0,0) ✗
x[0,0] ✗

	0	1	2	3
0	x[0][0]	x[0][1]		
1			x[1][2]	
2				x[2][3]

⊛ syntax:-

type Array name [No. of rows] [No. of Coloumns];
int x[3][4];

- intialize Array x Using User input.

```
int x[3][4], r, c;
for(r=0; r<3; r++)
{ for(c=0; c<4; c++)
    cin>>x[r][c]; }
```

r	c
0	0 1 2 3
1	0 1 2 3
2	0 1 2 3
3	0 1 2 3

نفس الشيء.

Scanned with CamScanner

* intialize 2D-Array Using initializer list:

int x[2][3] = { {1,5,6}, {7,9,3} }



* { {1,5}, {6,7}, {9,3} }

sublist ⇒ 3 rows syntax error.

3 rows مرفوض، مرات مرفوض size عشان

* لو بعض ال sublist اكتر من explicit ⇐ syntax error

③ int x[2][3] = {1,5,6,7,9,3}; ✓ 100% ZP

row by row يعني /

Ex: int x[3][2] = { {1}, {2}, {3} }

	0	1
0	1	0
1	2	0
2	3	0

* ⑤ Given the following Array:

int x[3][3] = {5,6,7,9,2,1};

Adel the value 5 to the elements in the secound row.

for (int i=0; i<3; i++)

x[1][i] += 5;

5	6	7
14	12	16
0	0	0

Scanned with CamScanner

Ex: Add the value 5 to the elements in the secound Column.

Sol: for(int i=0; i<3; i++) --> لازم نكتب
x[i][1] += 5;

Ex: Find the largest element in each row in the 2-D Array?

int x[3][3] = { 5, -2, 4, 3, 1, 13, 7, 15 };

	0	1	2
0	5	-2	4
1	3	1	13
2	7	15	0

i x

--> 5 r0

--> 13 r1

--> 15 r2

```
int max, r, i;
for(r=0; r<3; r++)
{ max = x[r][0];
  for(i=0; i<3; i++)
  {if( x[r][i] > max)
      max = x[r][i]; }
  Cout << max << "row" << r+1 << endl;
}
```

لو طلبنا ونريد نعرف ان هذا الرقم في اي صف نضيف متغير اضافي

⊛ Charachter Array :- Array whose elements which are type of Char.

C- String :- Null ('\0') terminateel Character Array.

Char Arr[5] = {'A', 'b', '2', 'd', 'e'}

Char Arr2[5] = {'A', 'b', '2', 'd', '\0'}
Null

↳ C.String ⇒ Null بدها

3. Char Arr3[5] = "Hi12"; String Array لـ

0	1	2	3	4
H	i	1	2	\0

تنعمل بشكل أوتوماتيكي بدل \0

⊛ إذا نقص (size) بحاجز زيادة رح يعمل بحاله Null بدل \0.

'A' → Charachter value A
"A" → // A + '\0'

String name;
char name[20];
char name2[ ] = "Ahmad"; //size=6

تنعمل بشكل أوتوماتيكي String, Charachter Array

* الفرق بين (int , char) من حيث الطباعة Array ← cin ، .

* int x[10];
cin>>x; → syntax error.
cout<<x ✓ Address.

char x[10];
✓ cin>>x;
x[5] = '!';
✓ cout<<x;

يجوز طباعة العلم Char لأنه نوعها.

program كلمة من User ؟ لنفترض بأنه دخل

0	1	2	3	4	5	6	7	8	9
p	r	o	g	r	!	m	\0		

يشتغل cin هون بشكل طبيعي وبكمل لحتى يشوف كلمة (NULL) .

Ex: char x[5] = {'1', ' ', '2', '\0'};
cout<<x; → 1 2

Ex: char x[5];
// input string with spaces.
cin.get(x,5);
User: 1 23
1 23
4+1 → NULL
cout<<x;

1		2	3	\0

✳ C-string manipulation functions:
[char Array].

[1] strcpy (S1, S2) → Copy S2 to S1.

Ex: Char a1 [6] = "Hello";
Char a2 [6];
strcpy (a2, a1);
Cout << a2;

Hello

Ex: Char x [10];
strcpy (x, "welcome");
Cout << x;

welcome.

[2] strcmp (S1, S2) ⇒ Compare
S1 < S2 → -ve val (-1 وغيرها)
S1 == S2 → 0
S1 > S2 → 1 (+ve)

Ex: Char a1 [ ] = "Hi";
Char a2 [ ] = "Hello";
Cout << strcmp (a1, a2);
Cout << strcmp (a1, "Hi");
Cout << strcmp (a1, "Hi All");

10
-1

Scanned with CamScanner

[3] strlen(s): → Length of s, exclude '\0'.

```
Char x[10] = "C++";
Cout << strlen(x);
```

3

	C-string Char Array	string
input	cin >> cin.get(s,m+1)	cin >> getline(cin,s)
output	cout << for loops	cout <<
Copy	strcpy(s1,s2)	s1 = s2 ;
Length	strlen(s1), without Null (Array لحتى)	s.length() (لحتى string)
Comparison	strcmp(s1,s2)	<, >, <=, !=, ==

(*) Aggregate operations:

Character Array:

1. cin >>
2. cout <<
3. Copy [Assignment].

```
Char a1[6] = "Hello";
b = a;  → ✗ syntax error.
```

Scanned with CamScanner

✱ Array of strings:

[1] string list [10];

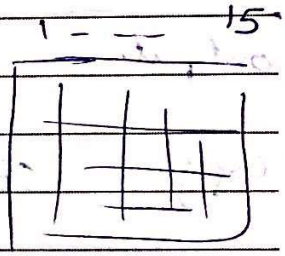
```
for(int i=0; i<10; i++)
    cin >> list[i]; ---> //
    getline(cin, list[i]); ---> //
```

بدون

[2] Array of string / c-string.

Ex: char list[10][16]; int i;

```
for(i=0; i<10; i++)
    cin.get(list[i], 16);
    strcpy(list[i], "snow white");
for(i=0; i<10; i++)
    cout << list[i] << endl;
```



"Chapter 6": functions:

① predefined functions → Compiler

→ pow(x, y)
→ sqrt(x)

→ cmath → (double)

② User Defind functions: "نعرفها"

✱ predefind functions:

[1] abs(x) → int
fabs(x) → float

⇒ absolute value

cmath

Scanned with CamScanner

زي صيغتي تقريب للأكبر.

[2] Ceil (x) ---- float ← تقريب ← cmath.

[3] floor (x) ---- float ← تقريب ← cmath.

[4] pow(x,y)
[5] sqrt(x,y) ⟹ cmath.

[6] fmod (x,y) ---- float ← integer باقي قسمة تقريبي cmath.

[7] $\log_{10}(x)$ ---- true ← small داخل cmath.

[8] islower (x) ---- int, char ← cctype
isupper (x) return true or false
---- true ← Capital داخل

Ex:

```
#include <cmath>
#include <cctype>
#include <iostream>
using namespace std;
void main ( )
{ int x=-5; float y=7.2; char c='B';
  cout<<abs(x-10);
  cout<<endl<<ceil(y)<<"\t"<<floor(y);
  cout<<endl<<sqrt(30+x);
  cout<<endl<<isupper(c);
  cout<<islower(103+x);
  cout<<endl<<islower('B');
}
```

15
8 7
5
1
1
0

Scanned with CamScanner

⊛ Function Call:

⊛ User defiend functions: → return value (string, int, bool, char, float, double)
→ no return value [void function]

⊛ syntax:

return value type.
↓
Function Type Function name (parameter list)
↓ function header

{
st1 ;
st2 ;
⇒ Function Body (function)
return expression ; → return value functions.
}

Function definition) st, Body) ، Header) ، ⊛

⊛ Examples:

1. Function that return the summation of two numbers.

function type. function name. parameter list.

header: `float sum (float num1, float num2)`

Body:

```
{
    return num1+num2;   or   float s = num1 + num2;
                             return s;
}
```

syntax error

```
void main ( )
{
    float x=3.5, y=-1.1;
    float s1;
    s1 = sum(x,y);
    cout<<s1;
    cout << sum(x+5, ++y);
    cout<<sum(3, 4.7);
}
```

function Call

function call: ينتقل التحكم إلى الدالة

main

2.4
8.4
77

Scanned with CamScanner

* إذا اختلفت في Header (الـ Header) عن الـ Header (الـ Body) ... Syntax error

* float sum (float num1, num2) → syntax error
لازم كل بارامتر له نوع البيانات الخاص به حتى لو نفس النوع.

* انتبه لـ "Comma" و "Comma separated list"

Ex: which of the following valid function Call ?

[1] float x = sum (floaty = 10, 15); ✗
(Header) Defintion الكلمة دي مش موجودة

[2] cout << sum (float x, float y); ✗
Defintion نفس الشيء

[3] cout << sum (float 5.3, 4); ✗
نفس الشيء

[4] cout << sum (5.1) ✗
2 parameters لازم

[5] cout << sum ("3.2", 4.1) ✗
string → Casting

[6] cout << sum (3, 1, 4, 1, 0.7) ✗
2 بس

[7] cout << sum ('A', 3.4); ✓

[8] cout << sum (1, 1, true); ✓

Casting

Scanned with CamScanner

predefined function.

* absolute function:

```
float abs (float n)
{ if(n<0)
    n*=-1;
    returnn;
}
void main ( )
{ cout << abs(-3.4);
  cout << abs (5.9); }
```

3.4
5.9

* power function:

```
int power (int x, int y)
{ int p=1;
  for(int i=1; i<=y; i++)
      p*=x;
      return p;
}
```

* طبعا لصير كتبت power او pow كاين جزء
* ما لصير راني بهاي الحالة نقول
#include <cmath>
لأنك استدعيت بالـ pow .

Scanned with CamScanner

⊛ Write a function isodd that returns true if it's argument is odd other wise return false.

⊛ Function prototype ⇒ Function Header ؛

Def function ؛ function ؛ main ؛

Sol: bool isodd (int val) ؛

```
void main ( )
{
  int v; cin>>v;
  if (isodd(v))
  cout<<v<<"odd";
    else
  cout<<v<<"Even";
}

bool isodd (int val)
{ if (val%2)
    return true;
    return false;
}
```

؛ else ؛ ؛

⊛ void function: No return value.

Ex: write a function that print the ascii code of a charachter.

⊛ معرفة في prototype أكتب، ثم لو ما أكتب عادي مرة الكود واقف.

Sol: void ascii (char); ---> syntax عشان

```
void main( )
{ ascii( );
  ascii( ); }

void ascii (char ch)
{ int a= ch;
  cout<< "ascii for"<<ch<<"is:"<<a
  <<endl;
  return; }
```

return لو طلبت int

Syntax error.

⊛ لو بدل return LCD, print the ascii code

```
int ascii (char ch)
{ return ch; }
```

Scanned with CamScanner

⋆ Peculiarities: الخصائص

1.
```
int secret (int x)
{ if (x>5)
  return 2*x; }
```
Code الكود ناقص

لو كان x أكبر من 5 هترجع 2*x

```
if (x>5)
  return 2*x;
  else
  return 3*x;
```
لو كانت قيمتها كذا

2.
```
int fun1 ( )
{ int x=25; // Local variable
  return 23, x; }
```
empty parameter list.

لو كان في أكتر من return هيرجع آخر واحد بس

```
void main ( )
{ cout<< fun1 ( ); }
```
return → 25

3.
```
int fun2 (int v)
{ int a=2, b=3;
  return 2*a+b, v+b; }
```
return

```
void main ( )
{ cout<< fun2(5); }
```
8

type ... ...
... ... function ... ...
function cast the return value:

```
[4] int sum (float a, float b)
    { return a+b ; }

    void main ( )
    { Cout<<sum (5.3, 7.4) ;
      Cout<<sum (5.1, 4.2) %2 ; --->
    }
```

12
1

4
4.5

bool 1
1

⊛ Reference & value parameters:

[1] Value parameters.

- formal parameters that receives a copy of the Content of Corresponding actual parameter.

- During program execution:-
formal parameter manipulate the data stored in it's own memory.

- Changes to formal parameter don't affect the actual parameter.

Scanned with CamScanner

[2] Reference parameters (&) :-
- formal parameter receives the Location (address) of the corrosponding actual parameter

- During program execution:
Changes to formal will change the actual parameter.

- both formal & actual parameter refer to the same memory location.

Are Useful in:-
[1] return more than one value.
[2] Change the actual parameter.
[3] save space.

formal → لا تستقبل / location
actual → لا تبعت / memory address.

Scanned with CamScanner

ignored by compailer.

Ex:
```
void f1(int);   -> value parameter
void f2(int&);  -> ref parameter.
void main( )
{ int a=5, b=5;
  f1(a);  // function call.
  f2(b);
  cout<<a<<"\t"<<b;
}
```
} ⇒ proto types.

inf1: 9
inf2: 9
5 9

ما يعبر عن قيمة cout

```
void f1(int x)
{ x+=4;
  cout<<"inf1:"<<x<<endl;
}
void f2(int &y)
{ y+=4;
  cout<<"inf2"<<y<<endl;
}
```

value

⊛ باختصار نعني بال ref إنه تعديل داخل ال void function وبالتالي عدم الحاجة لإعادة القيمة value وتغييرها (في ال global)

⊛ لو نسيت أول علامة فوق = syntax error لأنه بتبقى تعريف لل function.

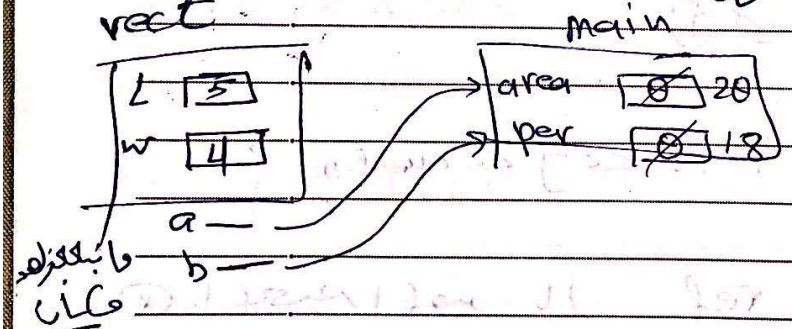
⊛ ال value تضعها زي قيمة وما تتغير.

Scanned with CamScanner

Return more than one value by ref parameters.

Ex: void rect (float &, float &);
void main ( )
{ float area=0, per=0;
rect (area, per);
Cout << area << per; }

void rect (float &a, float &b)
{ float L, W;
cin >> L >> W;
p = 2*L + 2*W;
return; }



5 4 ← input
20 18

(*) لو شلت & ref رح يطبع صفرين
الا اذا قيمها تغير من ref.

* reference parameters change the actual values:

```
void swap(float &n1, float &n2)
{ float temp = n1;
  n1 = n2;
  n2 = temp; }

void main ( )
{ float a=5.2, b=3;
  cout<<"a="<<a<<"b="<<b<<"Before swap";
  cout<<"\n";
  swap(a,b); // function Call
  cout<<"a="<<a<<"b="<<b<<"After
                                swap\n";
}
```

n1 = a
n2 = b

a = 5.2 b = 3 Before swap

a = 3 b = 5.2 After swap

* لو طبعت a في الـ main بعد الـ function هتطبع 5.2 ولا 3؟
هتطبع 3 لأن الـ ref غيّر قيمة a جوه الـ function

لو عايز تطبع a = 5.2 في الـ main لازم تعمل Cout قبل ما تقفل } بتاعة الـ main
فقط الـ ref يعدل القيم

Scanned with CamScanner

Storage Class & Scope Rules

⇒ Storage Class: determine the period during which the variable exist in memory.

[1] Automatic identifiers: Created [block entry] & destroyed (block exit) within it's block.
- inside any block ⇒ Local by default.
- intialized with rubbish.

[2] static identifiers: exist from the point of decleration for entire program execution
- out side any block → static by default.
- intialized with 0.

(*) Scope: where the identifier can be accessable in a program.

[1] Local identifier: within function or block scope.

→ static by default.

[2] Global identifier:
- file scope (outside functions).
- intialized with 0.

Scanned with CamScanner

⊛ Storage class and scope Rules:

الفترة الزمنية اللي بيفضل فيها المتغير بالذاكرة.

(1) Automatic: scope لكتلة التي تم تعريفه فيها
(2) static: يفضل موجود بالذاكرة طول فترة البرنامج (حتى بعد انتهاء main)

⊛ Global identifiers: static by default

من لحظة تعريفهم لنهاية الـ function.

automatic.

Ex: local variables within block scope:

```
void main ( )
{ int x = 3;        --> automatic local variable
  while( x<5)
  { int f=1;        --> automatic local variable
    x+=f;             لحد الـ while
    f++;
    cout<<x<<"\t"<<f<<endl;
  }
  cout<<x;          --> main هنا بالذاكرة صح لأنه متعرف فوق في الـ
  // cout<<f;       --> هنا بالذاكرة ماعاد تتنفذ cout لأنه ماعاد
}                     main موجود في الـ declaration
```

to main.	
4	2
5	2
5	

⊛ أي Variables بتتعرف فوق الـ main هتكون automatic فقط

⊛ لكن لما كنا نستعمل الـ loop وتوقف المتغير فقط داخل الـ loop.

Scanned with CamScanner

⊛ Static variables within block scope:

```
void main( )
{ int x=3;
  while(x<5)
  { static int f=1;
    x+=f;
    f++;
    cout<<x<<"\t"<<f<<endl;
  }
  cout<<x;
  //cout<<f;
}
```

x: 3 ← 4 ← 6
f: 1 ← 2 ← 3

static int f=1 ← يتم إنشاؤها مرة واحدة فقط. (أي أن البداية = 1 من المرة الثانية فصاعدًا)

main
x 3
while
f* 1

//cout<<f; ← ضغط بالماوس على cout تظل إبدأ، لأنه تعرفها فقط while داخل

⊛ الأهم: كل ما مر على static int f=1 فقط يتخطاها ولا يمر عليها مرة أخرى فقط. ما لم تعرفها (declaration) بالمرة الأولى فقط.

4	2
6	3
6	

⊛ هل يمكن لأكثر من دالة variable static، نعم لكل دالة مرة ومرة فقط.

Scanned with CamScanner

⊛ Local variables within function scope:

Ex:

```
int f ( )
{ int v=1;
  v+=5;
  return ++v; }
void main( )
{ int v1=3, v=0;
  cout<<f( );
  cout<<f( )<<endl;
  cout<<v1<<"\t"<<v;
}
```

77
3 0

لازم يكون فيه توافق بين ال function definition و function call

⊛ فائدته أنه ممكن يكون في كذا v فوق، و v تحت، طالما scope مش بعض.

error، لأنه كل واحد موجود لـ

[4] static variables within function scope:

Ex:

```
int f( )
{ static int v=1;
  v+=5;
  return ++v; }
void main( )
{ int v1=3, v=0;
  cout<<f( );
  cout<<f( );
  cout<<v1<<"\t"<<v; }
```

7 13
3 0

function call

ال static بتفضل موجودة لـ 1، لأنها بتتخزن في الـ data segment.

⊛ لو حاولت v فوق ال main إنك تستدعيها فوق عند ال func و كملت cout زي ما هي لـ main ~~for~~ لا، v مش معرفة

Syntax error.

File scope.

* Global variables :- static by default.
- declared out side functions
- intialize with zero(Local or global)

static → optional → initialize with zero

Ex: int x=7, y; // static global variables.
void f(void) { x++; y--; }

cout << x << y << endl; }

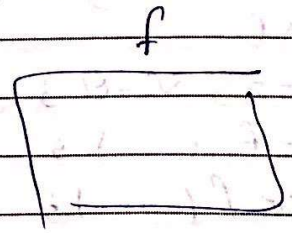
void main( )
{ int y=13, z; // automatic local variables
auto by default

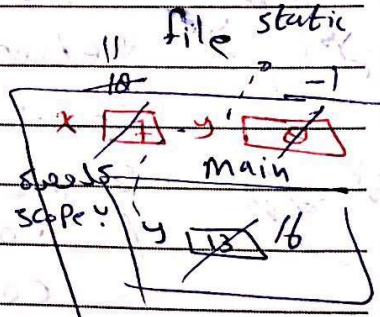
x+=3;
y+=3;
f( );
z=x+y;
cout << x << y << z << endl;
cout << ::y << endl; }

scope resolution operator.

#-1
11 16 27
-1

يستخدم لما يكون عندك قيمتين لواي وحدة local ووحدة تستخدم فقط مع global (global variables.





* أي global variables تعريفهم هو إذا مش بيقدر يعدل عليهم

* تذكر Intialize ← static الـ zero ، وإذا لقيت ...

main ... global ... slide 60,61 slide 64

file

Ex: float x=2.1, y=3.4; //s.G

```
void main ( )
{ float x=5.7; //a.L
  if (floor(x) >12)
    cout << ceil(x);
  else
  { float y=1.1;
    y++; cout << pow(floor(x), ceil(y)); }
    cout << x << y;
    cout << ::x << ::y; }
```

out of scope }.

x 2.1 y 3.4

main: x 5.7

else: y = 1.1 X 2.1

125
5.7 3.4
2.1 3.4

else ... static ...

(X) scope else ... global ... main ...

(X) ... else scope ... main ...

Ex: Local variable within block scope.

```
void main ( )
{ char c1='A', c2='a';
  ++c1;
  { char c2='B', c3='D';
    c2+=3;
    cout << c2 << c3 << endl;
    // cout << ::c2;
    cout << c1 << c2;
  // cout << c3; }
```

Block ...

c1 A B c2 a

c2 B c3 D

E.D
B 9

static ...

Scanned with CamScanner

prototype

```
Ex: void f (int x)
{ int z = x + y ; // cout << z ;
}
int x = 5 , y = 4 ;
void main ( )
{ f(6) ;
  cout << x << y ;
}
```

10
5 4

Global

⊛ function over Loading:

functions have the same name but different parameter list → types, order, Number

⊛ function segneture = function name + parameter list.

Ex: void f (int) ; → function segneture
int fun ( ) ; → // //

function name + parameter List

function over Load

functions

parameter List

Scanned with CamScanner

ignored by compailer.

Ex: void print (int y); } function prototype.
void print (char);
void print (float);

void main ( )
{ print (5);
print ('A');
print (7.3); }

→ function call.

void print (int x) → syntax error.
~~void~~ {cout << x << endl; }

void print (char x)
{cout << x << endl; }

void print (float f)
{cout << f << endl; }

5
A
7.3

* يمكن كتابة أكثر من function بنفس الاسم overload بشرط أن يكون parameter list مختلف.

* داخل الـ function لو مش هتعمل x، كتابة x/ header

لأنه كل مرة بتشتغل بـ scope جديد.

* الـ function اللي بتعرف الـ function اللي بتستدعيها (function call) لازم يعرف parameter List بتاعتها (عدد وترتيب وأنواع) float, int, وهكذا.

Scanned with CamScanner

Over Loading → نفس الاسم

int x = 20;

Ex: ① void f(int x)
{ x += 10;
cout << x << endl; }

② int f ( )
{ return x++; }

③ int f(int a, int b)
{ return a+b+x; }

④ // void f ( ) { cout << x << endl; }

void main ( )
{ f(7); cout << endl;
cout << f(3, 5.4) << endl;
cout << f ( ); }
cout << x;

17
28
20
21

كل متغير له قيمة داخل الـ global تم التعريف عليها من الأعلى

* ملاحظة مهمة: التعريف على global يعني أن المتغير متاح للكل أو كل دوال البرنامج إلا إذا تم تعريف متغير محلي بنفس الاسم داخل دالة معينة.

* ملاحظة: لو كانت Comment أو كانت في الكود نتجاهلها لأن الـ Comment لا تؤثر على الكود، 2 functions لازم يختلفوا بالـ parameter list وإذا كانوا نفس الـ parameter list لازم syntax error.

* لاحظ أن الـ function رقم ③ أخذت قيمتين وقررت فهم أنهم float وعملت casting لكل float لأن الـ f.D نفسه كامل بأنه يستقبل 2 integers.

* لو كان بالبرنامج أكثر من functions ليستقبل كل واحد منهم float أو integer أو char كان ما يلاحظ عمل Casting.

Scanned with CamScanner

⊛ Default Arguments:

```
void f(int x)
{ cout<<x+5; }

void main( )
{ f(5);
  f('A');
  //f( );
  //f(5,6);
}
```

فقط يستقبل رقم

Default Arguments يكتب في الـ prototype فقط (لو موجود)

Ex:
```
void fun (int a=3, int b=7, int c=1);
void main( )
{ fun (9, 4, 10);
  fun (4, 5);
  fun (8);
  fun ( ); }

void fun (int a, int b, int c)
{ int w = a/b + (++c);
  cout << w << endl; }
```

13
2
3
2

الأوفر دائمًا يأخذ x فقط، أو يأخذ x و y فقط، أو يأخذ x و y و z، أو لا يأخذ شيئًا، وفي هذه الحالة يأخذ default

⊛ حالة لو الأول لا، إذا الـ User دخل قيم، تتغير تلقائيًا هذه القيم إلى الـ void fun

⊛ فقط: لو بالك إنك رح تكتب هذه القيم بالغلط في الـ prototype، رح تنعمل عندك syntax error

⊛

Default Arguments

Ex:
```
bool f ( int a , char c='B')
{ if (a==c)
   return true;
  else
   return false; }
void main ( )
{ cout<<f(100,'d')<<endl;
  cout<<f(99);
// cout<<f ( ); }
```

1
0

هاد الكود خطأ لازم يكون الأول أخذ قيمة وحدة ل a

⊛ زي ما كنا بال default fefa, انت بكبر بال prototype فقط, كمان تكتبها بال Header غلط Header بتكتب بال الحدد.

⊛ داتيا بال Default بنحط بالآخر بحيث, لقيم الأولية ترجع للي قبلها, إنه default و الأشي صعوبته قيم بآخر بال default (syntax error)

⊛ ال function لويلي إياه وال جوى:
الأمثلة

```
{ int v; char c;
  cin>>v>>c;
  if (f(v,c))
cout<<v<<" is ascii for: "<<c<<endl;
  else
cout<<v<<" is not ascii";
```

Scanned with CamScanner

default القيمة هي ؟ A( ) → function call في حال لم يتم تمرير قيمة

zero

Ex: int A(int =0);
void A(int, char);

which of the following prototype can be overload with these functions ?

① int A(char); ✓
② int A(string, int); ✓
③ int A( ); ✗ syntax error.
④ ~~int~~ float A(int); ✗ syntax error.

⊛ Recursive functions:
- is the function that call it self يستدعي نفسه

⇒ Base Case (Condition) :- when this condition is met the function stops calling it self.

⇒ Recursive Call :- each time are Cursive Call is made it comes closer to the base case.

⊛ Recursive Call يتغير باستمرار ← Base Case الـ

⊛ Base Case هو الحالة التي يتوقف عندها b/ ؟.

Scanned with CamScanner

fact(5) = 5 * fact(4)

4 * fact(3)

3 * fact(2)

2 * fact(1)

1 * fact(0) = 1

Ex:

```
int fact(int n)
{ if(n<=1)
    return 1;        ] ⇒ Base case
  else
  return n*fact(n-1); → recursive Call
}
void main( )
{ cout<<fact(5); }
```

120

كل دالة فيها Loop يمكن تحويلها لـ Recursive

Base Case لازم يكون فيها

Ex: ① Adding up Numbers from 1 to any given number.

Sol:

```
int sum(int n)
{ If(n==0)
    return 0;
  else
  n+sum(--n); }
void main( )
{ cout<<sum(5); }
```

15

5 + sum 4

4 + sum 3

3 + sum 2

2 + sum 1

1 + sum 0

الإجابة / 15

Scanned with CamScanner

Ex: print numbers from n to 1:

```
void print_dow(int n)
{ if (n==1)
  cout<<n<<endl; return;
  else
  { cout<<n<<endl;
    prind_down(--n); } }
void main ( )
{print_down(5);}
```

5
4
3
2
1

Ex(4): print numbers from 1 to n:-

```
void print_up (int n)
{ If (n==1) cout<<n<<endl;
  return;
  {print_up (--n);
   cout <<n<<endl; }
void main ( )
{ print_up (5); }
```

print_up(5)
print_up(4)
print_up(3)
print_up(2)
print_up(1)

1
2
3
4
5

(*) من يوجد cout ، الخط يطبع.

Scanned with CamScanner

	start	end
ispalindrome 0		5
ispalindrome 1		4
ispalindrome 3		2

Ex: check whether a number is prime:

```
bool isprime(int n, int i=2)
{ if (i==n || n==-1)
    return true;
  if(n%i==0)
    return false;
  return isprime(n, i+1); }
```

*

Ex: (slide 77):

```
bool ispalindrome(int start, int end, string str)
{ if(start-end>=0)
    returne true;
  if(str[start]!=str[end])
    return false;
  return ispalindrome(start+1, end-1, str); }

void main( )
{ string str="nurses run";
  cout<<ispalindrome(0, str.Length()-1, str); }
```

abccba ⇒ palindrome

abcbca ⇒ not palindrome.

(*) str.Length ⇒ return the number of character in the string without NULL.

start = 0, end = str.Length()-1

Scanned with CamScanner

⊛ Arrays as parameter to functions.

- Array are passed by refrence only.
- size of the array ignored by the compiler
  ← يتجاهل البرنامج.

Ex:
```
void Update (int A[10], int size, int val)
{for(int i=0; i<size; i++)
  A[i] += val; }

void main ( )
{ int Ar[5] = {2,3,4,-1,7} ;
  Update (Ar, 5, 3) ;
}
```

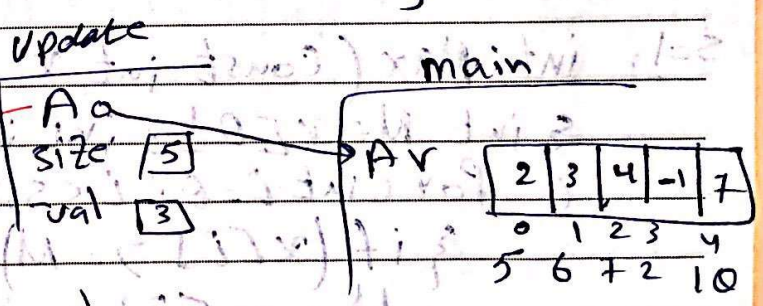
Array name ← / size ← / Update value function Call

```
for(int i=0; i<5; i++)
  cout << Ar[i] << "\t"; }
```

out put

5 6 7 2 10

Ar ← A ← Update



⊛ Syntax (sub ;)

[1] Update (Ar[5], 5, 3) ✗ ← لا يتم التعريف بهذه الطريقة

[2] Update (Ar[ ], 5, 3) ✗ ← لابد أن يتم تحديد حجم الـ Array ;

Scanned with CamScanner

④ Constant Arrays as formal parameter

→ to prevent a function from changing the actual parameter.

Ex: print Array elements.

```
void main (const int A[ ], int size)
{ for (int i=0; i<size; i++)
  cout<<A[i]<<"\t";
  cout<<endl; }

void main ( )
{ const int s=7; int v[5]={2,4,5,8,3};
  print(v,s); }
```

Ex: Write a function that finds the largest element in array.

```
Sol: int Max (const int x[ ], int size)
{ int M=x[0]; int index=0;
  for (int i=0; i<size; i++)
  { if (x[i]>=M)
    M=x[i]; } index i
  return M;
}
```

→ index

the Largest element

index of Array

⑤ و ن ا و ل

Ex: Const int size = 5;
int search (int x[ ], int val)
{ int ctr = 0;
for(int i=0; i<size; i++)
if (x[i] == val)
ctr++;
return ctr; }

Passing 2D Array to function: --> ignoreal.

void print(const int x[5][3], int rows)
{ for(int r=0; r<rows; r++)
{ for (int c=0; c<3; c++)
cout << x[r][c] << "\t";
cout << endl; }

1 2 4
7 9 13
12 0 0

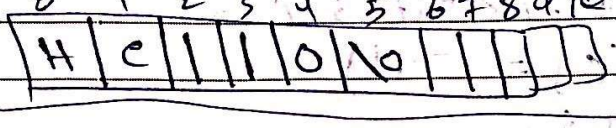
void main( )
{ int A[3][3] = {1,2,4,7,9,13,12};
print(A,3); }

int x[2][3];
int x[ ][3] →
↳ # of rows = ? ↳ # of cloums

C-string

Char Array:

```
Char x[10] = "Hello";
cout << x;
```



Hello

```
for(int i=0; i<10; i++)
{ if x[i] == '\0' break;
  cout << x[i]; }
```

```
for( int i=0; x[i] != '\0'; i++)
    cout << x[i];
```

Chapter 12: pointers

pointer = variable : memory location that Contains memory access.

pointer Declaration:

Syntax:-

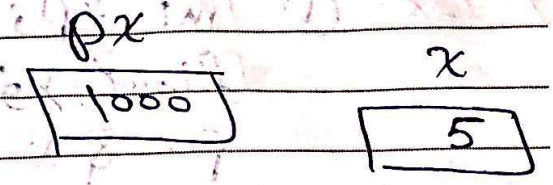
```
Type *pointerName;
```

Ex:
```
int *p1;
Char *p2;
douple *p3;
```

Scanned with CamScanner

address

Ex: int x=5;
int *px;
px = &x;
address ←



(*) Pointers Initialization:
- initialized with memory address or NULL → points to nothing

- Use & to get address of a variable.

Ex: Char x = 'A'; ← Capital
Char *px = NULL; == Char *px = 0;
Chary, *py = 0; points to nothing.
Address ← px = &x;
py = &y;
value ← (*px)++; // x++
*py = *px + 2;
Cout << x << "\t" << y

B	D

Scanned with CamScanner

(*) Ex: (1) int *p;
float x;
p = &x ✗

(2) float f1, f2;
float *pf1 = &f1; →
float *pf2;
*pf2 = &f2; ✗ → pf2 = &f2;

value ←→ address

(3) int x1, x2, *p1, *p2;
p1 = &x2;
p2 = x1; ✗ → p2 = &x1;

(4) int x, *px = &x;
int y = 5;
*px = y;
// px = y ✗



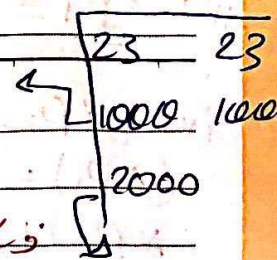
y 5

(*) x & int عبارة عن pointer
int Data عبارة عن
pointer Data

Ex: int r = 20;
int *p = &r; --> int *p; p = &r;

given
The address of r is 1000 & p is 2000

r++;
*p++;
cout << r << "\t" << *p << endl;
cout << p << "\t" << &r << endl;
cout << &p;



p كنوان ال

int*p int * p int*p
وبيعتبروا كلهم white spaces

cout << p; كنوان variable عشان كده

p ال

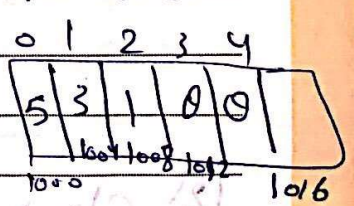
(*) int *p1 = NULL;
int *p2 = 0;
points to nothing لا يؤشر على أي شيء ولا يوجد له كنوان

int *p3 = 5; ✗
addd value.

arr = &arr[0]

(*) The relationship between pointer and Arrays.
→ Array name is a constant pointer that contains the address of the first array element
(كنوان لأول عنصر موجود في Array)
- pointers can do subscripting operations.

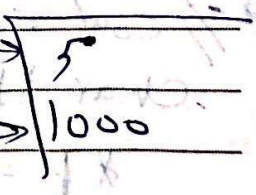
Ex: int x[5] = {5, 3, 1};



Const
→ x [1000] كنوان تبتدي

أول عنصر وعشان يتغير

cout << x; → Address of x[0]
cout << *x; ⟹ cout << x[0]



cin >> x; ✗ syntax error.

cout << x char ال Array/ طالما مش عنصر
مشروع يطبع عناصر الـ

Ex: int a[5] = {13, -2, 1, 4};

int *p1 = a; → address of a[0] (const pointer)

= int *p1 = &a[0];

int *p2 = &a[2]; → a[2];

(*p1) += 5;

(*p2) += 10;

for (int i = 0; i < 5; i++)

cout << a[i] << "\t";

cout << *(a+i) << "\t";

cout << p1[i] << "\t";

18	-2	11	40

5 مرات

اسم الـ pointer هو عنوان أول عنصر موجود في الـ Array

	0	1	2	3	4
18	13	-2	1	4	0

p1[0] = p2[0]

⊛ a[0] ≡ *a ≡ *p ≡ p[0]

⊛ a[1] ≡ *(a+1) ≡ p[1] ≡ *(p+1)

↳ *a+1 = 13+1 = 14

⊛ a[2] ≡ *(a+2) ≡ p[2] ≡ *(p+2)

⊛ لأنه ممكن نستخدم الـ pointer زي الـ Array وبالعكس

[2] char c[4] = {'A', 'B', 'C', 'D'};

// cout << c; syntax error

char *p = &c[3]; → عنوان

*p = '?';

(p-1) = '';

for (int i = 0; i < 4; i++)

cout << c[i];

c+3

	0	1	2	3
c[]	A	B	C	D

p1

A B * ?

Scanned with CamScanner

⊗ مـ امثال، بـ بقى تذكر، فـ Rubbish يطبع لـ NULL

ماتقدر لأنه Cout توقف لكن NULL فقط بتدوقاع لـ Charachter Array // Cout << C ؛ ←

~~لكن تنفع بـ C-string NULL~~

```
double A[5] = {0};
double *p = A; ----> double *p = &A[0]
*p = 10.1;
p++;
(*p)++;
p+ = 2;
*p = 14.3;
--p;
*p = 9.4;
//A++;
for(int i=0; i<5; i++)
  Cout<<A[i]<<"\t";
```

10.1	1	9.4	14.3	0

⊗ Constant Array لـ

⊗ what is the Last address of p? &A[2]

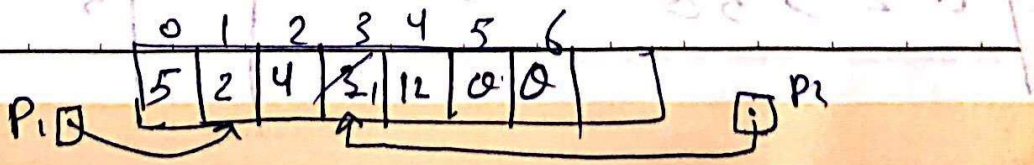
وقيمة لا يوجد

⊗ بالغا بيوك لـ Pointer لقدام (++) أو لورا (--) حتربك Logical error أي Array لـ خارج

لأنه عنوان P_1 أقل من عنوان P_2 ← if($P_1 > P_2$) ⇒ false

```
int A[7] = {5, 2, 4, 3, 12};
int *P1 = A, *P2 = &A[6], *P3 = 0;
P1++; P2--=3
if (P2 >= P1) *P3 -= 4; else *P1 = -10;
  if (P3 == NULL)
    Cout<<*(P1+2);
```

-1



Scanned with CamScanner

⊛ Functions & pointers:

~~Ex~~ void f(int*); → ignored

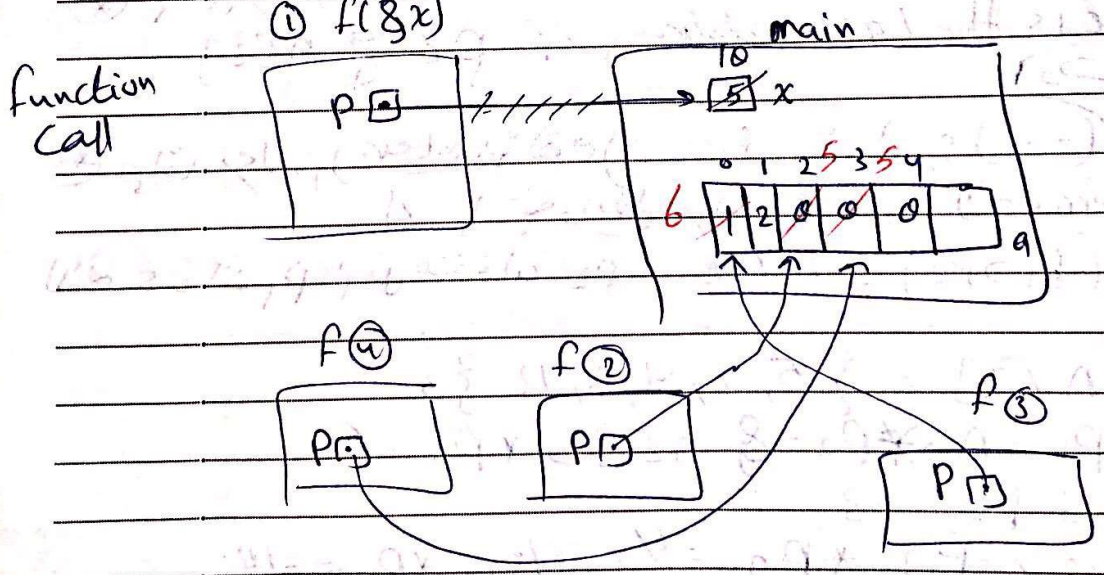
```
void main( )
{ int x = 5, a[5] = {1,2}; *ptr = 0; }
        ptr = &a[3]; }
```

① f(&x); ⇒ f(x) ✓
② f(&a[2]); ⇒ f(a[2]); ✗
③ f(a); ⇒ Array إرسال أول عنصر في
④ f(ptr); ⇒ f(*ptr) ✓ وهو المؤشر عليه

void f (int * p) ⇒ white spaces ✓
{ *p = 5; }

⊛ يرسل إلى الدالة أول عنصر في Array أو عنوان أول عنصر في Array
f(a); = f(&a[0]); أو f(&a); ✗ f(a[0]) ✓

⊛ what is the final value of x? x=5



⊛ what is the final value? x=5

6 2 3 5 0	*ptr. طلع
في حال كان العنصر في Array	5

Scanned with CamScanner

model 12

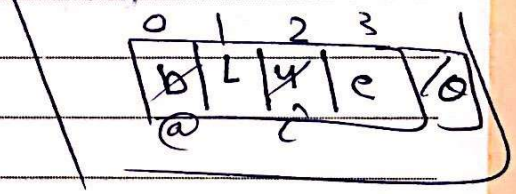
* Ex: void fun (char , char & , char *);
void main ( )
{ char a[10] = "blue";
fun(a[1] , a[2] , a);
value, ref, pointer
int i=0;
while (a[i] != '\0')
{ cout << a[i]; i++; } }

متغيرات طبع Array ال فقط وطبع مع Rubbish.

void fun (char a, char & b, char *ptr)
{ a='x'; b='i';
*ptr = '@';

@Lie

main



* ملاحظات: a[i] يعبر أيضاً عن a ; a[1] → a , a[2] → b
Const Pointer
a → pointer

* إذا بدي أبعت كامل Array راجع by ref
* لازم كتبة بعت كمان ال Array ref or value

* ال swap لازم by ref.

Ex: void swap (int *p1, int *p2)

{ int *temp = p1; p1 = p2; p2 = temp;

cout << "In swap" << *p1 << "\t" << *p2 << endl; }

void main ( )

{ int x = 1, y = 2;

int *px = &x; *py = &y; Declare + Initialize

cout << "Before swap" << *px << "\t" << *py << "\n";

swap (px, py);

cout << "After swap: " << *px << "\t" << *py << endl;

تبديل swap

تنفيذ دوال function ما تتأثر بالمتغيرات داخل main الـ

out put

Before swap 1 2

In swap 2 1

After swap 1 2

Scanned with CamScanner