



تقدم لجنة EiCoM الاكاديمية

تلخيص لمادة:

برمجة الحاسوب

جزيل الشكر للطالبة:

مروى حلوة



مراجعة ++C

(للمهندسة)

إعداد الحالبة

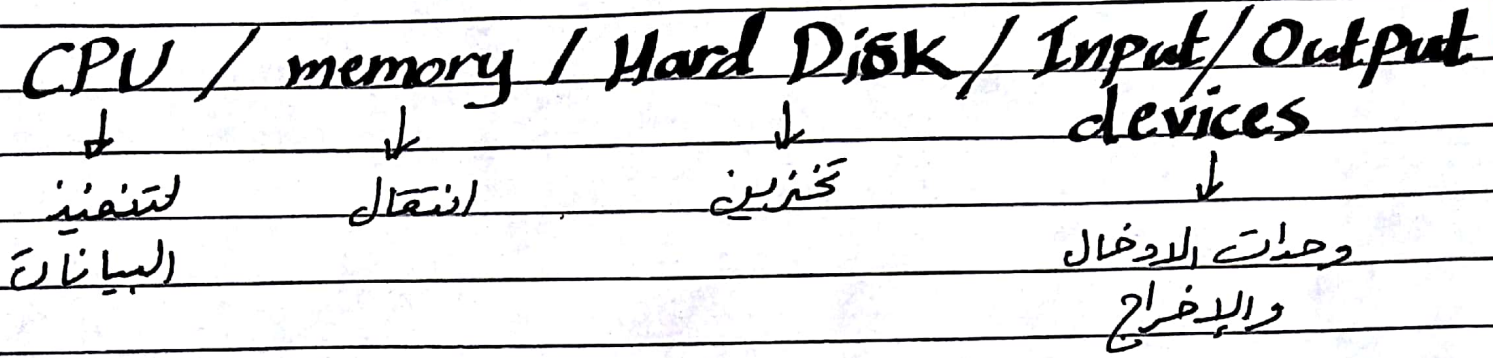
مروه حلوه

(دفتـر المهندسة
الاء ابوسرحان)

* Introduction :-

لبدء التعامل من البرمجة (software) يجب التعرف على كيفية انتقال البيانات داخل أجزاء الحاسوب .

* المصبرات التابعة لحركات البيانات :-



* Processing a C++ Program :-

① Editor :- (المحرر)

* تقوم فيه على كتابة ال code والتعديل عليه من زيادة أو نقصان . (في ال HD)

② Preprocessor :- (المعالج)

* يبدأ على المعالج مع (*)

* وكأنه مكتبة كبيرة تحوي معنى الكلمات والاشارات المستخدمة في كتابة البرنامج ليفهم الحاسوب معنى ما يتم كتابته . (في ال HD)

③ Compiler :- (المترجم)

* يجعل المترجم مهمتين رئيسيتين وهما :- (في ال HD)

① انه يتأكد من عدم وجود أخطاء كتابية في البرنامج

② انه يتحقق من لغة الانسان إلى لغة الآلة

④ linker :- (الربط)

* يجعل على تحويل ال code من لغة كتابة إلى برنامج تنفيذي (في ال HD)

⑤ Loader :- (المحمل)

* يعمل على ارسال البرنامج التنفيذي الى الذاكرة الرئيسية (main memory) .
(HD → memory) .

⑥ Execute :- (التنفيذ)

* وتعتبر الخطوة الأخيرة وهي تنفيذ البرنامج وإظهار النتائج النهائية .

* Program development :-

(معلومات عامة)

* لكتابة برنامج يجب على المبرمج (Programmer) ان يفهم جيداً ماذا يعمل البرنامج (يعني يتو بتستفيد منو)

* يجب عليه تحليل البيانات (يعني يعرف كم قيمة يح تدخل مثلاً وكم قيمة يح تخرج من البرنامج . مع بعض يح تدخل قيمتين وتطلع قيمة وحدة وهكذا)

* يجب عليه تحديد الشروط الموضوعية على البيانات (يعني مثلاً لازم الرقم الأول أكبر من 5 أو يقبل القيمة على كذا وهكذا)

* نختار اسرع طريقة لكتابة البرنامج بأقل عدد أسطر اذا أمكن (لأن زيادة عدد الأسطر يعني زيادة سرعة البرنامج)

* هاد الكود يشغل عالم عن البرمجة والهدف دائماً اخفض النتائج بأقل تكلفة وأقصر وقت .

* البرمجة هي تحويل الجهل من لغة الانسان الى لغة الآلة .

* جهاز الحاسوب عبارة عن (شغيل) غير قادر على اتخاذ القرار .

* المبرمج زي ما يكتب الحاسوب تنفيذ و لازم يفهم نتعامل معاه بطريقة صحيحة .

* Notes :-

ادخال
Input

اخراج
Output

```
#include <ostream >
```

①

عليه الاستغناء
للمتغيرات وبيان
معالجة البرنامج

يعني

* عند البرنامج من المخرجات
والمخرجات لبيان كتابة
ال code

2

using namespace std [و]

تستخدم لاستدعاء جزء معين من المكتبة لتفادي سوء التمازج .

3

```
int main
{
```

```
return 0; }
```

هو عبارة عن الدالة الرئيسية التي تحتوي على البرنامج التنفيذي ، يكتب كل شيء داخل هذه الأقواس لتفادي

4

cin, cout

هنا لها تيجي مثلا تخرج اسمي على شاشة المخرجات ، ف من المخرجات نخرج مثلا print 523 هاهي لغة السير ، لها آخري انسخها على البرمجية ليكن بدل print + cout و ينجها بامثلة اخضر من (<<) مرتين ، جديده بوقت وكي بين اياه بين (=) و آخر اسمي كذا (ف) فاصلة منقوطة . بالنسبة ل cin ، بتستخدمها لما بينا لتستخدم (user) يعطينا قيمة كغير نستخدمها داخل البرنامج . ال cin بتستخدم بوجه البرمجة (>>)

5

كيفية عمل البرنامج كالاتي ، عبارة عن تعريف متغيرات واعطاه قيم وادخال هذه المتغيرات فيه عمليات حسابية (+, -, *) او منكمية (<, >, =) ، واعطاء النواتج المطلوبة (كل ما د كتبه عثمان رقم و الامثلة بتوضع معنا)

6

نهاية اكلمة بالفاصلة المنقوطة وليس بسطر جديد

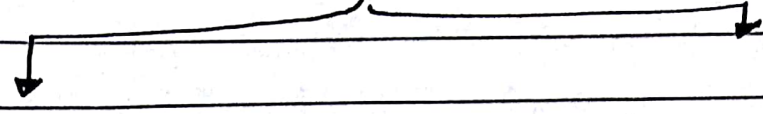
7

```
cout << "ahmad" << endl; ← لا نزال المخرجات سطر جديد
```

```
cout << "ahmad\n";
```


* أنواع الـ Errors

تنقسم إلى نوعين



syntax error

* يكون الخطأ عند كتابة الـ code

مثل عدم كتابة الفاصلة المنقوطة

```
int x = 5 □ x
```

أو استخدام كلمة معروفة كأسم

متغير

* لا تخرج نتائج على شاشة المطرمان

Logical error

يكون الخطأ منطقي

يكون من النتيجة التي يري

أيضا تخرج مثل عدم اعطاء

المتغير قيمة واستخدامه في

الـ code مسبقاً

```
int x, y, sum;
```

```
sum = x + y;
```

↳ Logical error

* تعريف المتغيرات :-

طريقة التعريف :- ①

* لتعريف متغير نستخدم ما يسمى بـ Data type وهي نوع المتغير

الذي سنستخدمه ، صحيح ، عشري وغيره ، المتغير يستخدم

أحياناً للاحتفاظ بالقيم داخله وتنقل القيم داخل البرنامج دون

تغييرها .

* الشكل الأساسي للتعريف ← data type variablename

* لا يمكن تعريف متغير مرتين (syntax error)

* الأسماء القصيرة والصغيرة تعتبر متغيرات مختلفة

```
int I = 2;
```

→ syntax error .

② Data Types :-

- ① int (4 byte) يستخدم لتخزين الأرقام الصحيحة
- ② float (4 byte) يستخدم لتخزين الأرقام العشرية والعشيرة
- ③ double (8 byte) نفس استخدام float لكن الفرق في الدقة
- ④ bool (1 byte) بحزنتيه (0, 1) أو (true, false)
- ⑤ char (1 byte) يحزن فيه أي قيمة على الأسيبورد
واحدة فقط

1 byte = 8 bit

Notes on data types :-

char نوع a فلا إذا تم فرزها من قبل المبرمج يعيدها

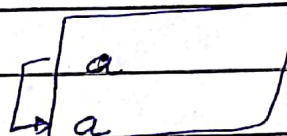
char x = 'a';

x = 'a' أو x = 97

char y = 97;

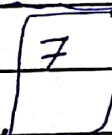
cout << x << endl;

cout << y << endl;



char x = '97';

cout << x;

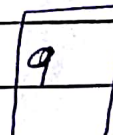


char نوع آخر فإنه إذا تم إعطاها قيمتها من قبل المبرمج

char x;

cin >> x; // 97

cout << x;

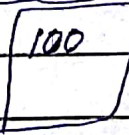


char نوع أول فإنه إذا تم إدخال قيمتها من قبل المستخدم

أي من الأرقام من 0 إلى 255

char x = a;

cout << x + 3;



إذا أدخلنا متغير من نوع char على قيمة حسابية فإنه يعامل مع قيمة

من الأسس كود (a = 97 ويزيد واحد b = 98 ويزيد واحد)

(A = 65 ويزيد واحد B = 66 ويزيد واحد)

bool x = True;

(syntax error)

المتغير من نوع bool ليس است

قيمة true أو false يجب أن تبدأ

أن أول حرف (small)

المتغير bool يعبر أي قيمة = 1 ما عدا false و 0 يعبرها 0 (الحزنتيه)

* اخطاء في هذا البرنامج ان a هي متغيرة
 داخل () فاعتبرها البرنامج variable
 $char x = a;$ $cout << x;$ \rightarrow syntax error
 syntax error

③ شروط كتابة variable :-

- ① ان لا يبدأ برقم \rightarrow $int 3x = 5;$ X
- ② ان لا يكون كلمة محجوزة
الجدول المحجوزة Small \rightarrow $int for = 4;$ X
- ③ لا يوجد فراغ (white space) \rightarrow $int Float = 2;$ ✓
- ④ لا يحوي على (+ , - , * , /) \rightarrow $int x + = 2;$ X
- ⑤ لا يحوي على (# , @ , * , -) \rightarrow $int x $ = 5;$ X
- ⑥ (underscore character) \rightarrow $int x _ 3 - 4;$ ✓

Note :- // تعني comment وليس تنقيح و سطر واحد \rightarrow //

/*
*/
بأكثر من سطر

* coding example :-

الكتب برنامج لتعريف 3 متغيرات و طباعة المجموع والقليل :-

```

#include <iostream>
using namespace std;
int main( )
{
  int x, y, z;
  x = 3; y = 4; z = 1;
  int sum = x + y + z;
  double ave = sum / 3;  $\rightarrow$  استعملنا double لان الناتج اعلى لا يساوي
  cout << ave << sum << endl;  $\rightarrow$  عدد صحيح بل هو عشري
  return 0;
}

```


ex :-

- | | | | |
|--|--|--|---------------|
| ① <code>bool x = true;</code>
<code>cout << x;</code> | | ② <code>bool x = 100;</code>
<code>cout << x;</code> | |
| ③ <code>bool x = 'a';</code>
<code>cout << x;</code> | | ④ <code>bool x = a;</code>
<code>cout << x</code> | syntax error |
| ⑤ <code>bool x = "a";</code>
<code>cout << x</code> | | ⑥ <code>double d3;</code>
<code>d3 = d3 + 1;</code>
<code>cout << d3;</code> | logical error |

coding ex :-

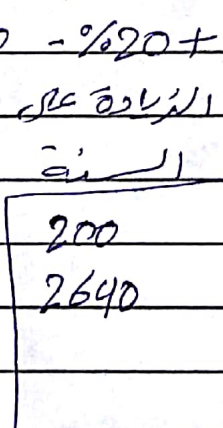
Write a program contains the value of the monthly salary of an employee and his salary is 200 \$. Print his monthly and annual salary with a premium of 20% and a discount of 10%.

نكتب البرنامج لحساب الراتب السنوي والاربعيني مع زيادة 20% وخصم 10% من الراتب الشهري.

```

#include <iostream>
using namespace std;
int main ( )
{
    int x = 200;
    int y = x * 12;
    int z = y * 0.2; int y = y + z;
    cout << x << endl;
    cout << y << endl;
}

```



x	y	z
200	2400	240
	2640	

★ Casting :-

وهو تغيير قيمة المتغير من اجهزة تلقائياً وعلى اقله قسماً

```

① int a = 100;
   bool b = (bool)a;
   bool c = bool(a);
   cout << b << "\n" << c << endl;
   cout << a;

```

int	bool	bool
a	b	c
100	1	1

→ حرة وبيع الأرقام

```

② float x = 2.5;
   cout << int(x) << endl;
   cout << x;

```

	x
2	2.5
2.5	

• كل casting داخل آلة cout لا يغير من قيمتها الأصلية.

```

③ int a = 7.666;
   cout << float(a) << endl;
   cout << a;

```

int
7
7

• كل casting في البراية في ما يرجع يعيد القيمة الأصلية

```

④ float a = 7.666;
   cout << int(a) << endl;
   cout << a;

```

float
a
7.666
7
7.666

• الجملة المعجزة لكل ال casting :-

Static cast < datatype > (variable);

```

⑤ int x = 10;
   bool y = static_cast<bool>(x);
   cout << x << "\n" << y;

```

int	bool
x	y
10	1
10	

```

⑥ char x = 97;
   char y = 'a';
   cout << int(y) << endl << x << "\n";

```

char	char
x	y
97	a
a	a

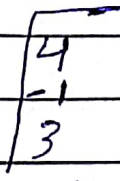
* C++ Arithmetic Operations :-

* العمليات الحسابية في برمجة C++ :-

```

① int x = 3;
   int y = 2;
   cout << x+1 << endl;
   cout << y-x << endl << x << "\n";

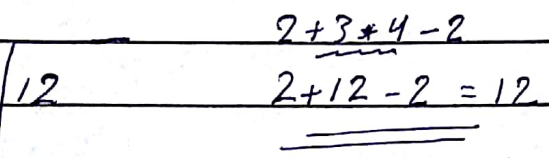
```



```

② int x=2, y=3, z=4;
   int result = x+y*z-x;
   cout << result;

```



* حالات القسمة العادية :-

```

③ int a=5;
   int b=2;
   cout << a/b;

```

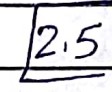


① إذا كان ناتج طباعة مباشرة :-
* إذا كان المتغيرين int ← int

```

④ int a = 5;
   float b = 2;
   cout << a/b;

```



* إذا كان احد المتغيرات على الأقل float
float ← float

⑤ إذا تم تخزين ناتج الطباعة داخل متغير ثابته :-

```

⑤ int a=5, b=2;
   float c = a/b;
   cout << c;

```



* إذا كان كلا المتغيرين int وحزن داخل
float ← int

```

⑥ int a=5, b=2;
   int c = a/b;
   cout << c;

```



* إذا كان كلا المتغيرين int وحزن داخل
int ← int

```

⑦ float x = 5;
   int y = 2;
   float c = x/y;
   cout << c;

```

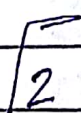


* إذا كان احد المتغيرات على الأقل float وحزن داخل
float ← float

```

⑧ float x=5; int y=2;
   int c = x/y; cout << c;

```



* إذا كان احد المتغيرات float (على الأقل) وحزن داخل
int ← int

* بافتی القسمة و حالاته :-

بافتی القسمة أو mod أو % يستخدم لطباعة ما تبقى
عن الأرقام عند قسمة رقمين من نوع int .

* ما تبقى من قسمة الرقم 9 على الرقم 4

```

(9) int x=9;
    int y=4;
    cout << x%y;
    
```

* بما أن الرقم الأول أصغر من الرقم الثاني

```

(10) int x=4;
    int y=9;
    cout << x%y;
    
```

منه لم تقبل القسمة (نتيجة القسمة العادية = 0) و بافتی القسمة هي الرقم الأول كامل .

* يجب ان يكون كلا الرقمين من نوع

```

(11) float x=9;
    int y=4;
    cout << x%y;
    
```

int ← Syntax error (مهمة جداً جداً)

* يمكن استخدامها إذا

```

(12) int x=2;
    float y=4.5;
    cout << x << y << endl;
    cout << x%int(y) << endl;
    
```

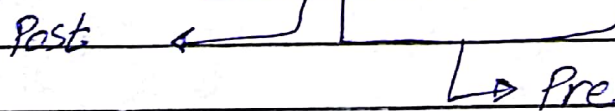
قنا بال casting + cout ، لا تؤثر على القيم الأصلية

```

(13) int x=2;
    x+=2;
    cout << x;
    cout << x+2;
    cout << x;
    
```

$x+=2 \Rightarrow x=x+2$ *
 $x++ \Rightarrow x=x+1$

Post and Pre increment/decrement *
($x++$, $++x$, $x--$, $--x$)



* عملية زيادة أو نقصان من قيمة المتغير ← وضعت هذه
الفترة لأن شكلها جديد وليست صيغة ابدأ ←

$x++ \Rightarrow x=x+1$ * $x++ \Rightarrow$ اطبع ثم زد
 $++x \Rightarrow$ زد ثم اطبع

* جميع العمليات الحسابية
 الموجودة في لغة C تؤثر على قيمة المتغير
 ما عدا post and pre تغير
 على قيمة المتغير .

⑭ int x;	x	
x = 2;	2	4
x++;	3	6
++x;	4	
cout << x++ << "\n";	5	
cout << ++x << endl;	6	

* ترتيب أولويات العمليات

- ① ()
- ② Post increment / decrement
- ③ ! → not x = 5 → true !x → 0 $\frac{x}{x}$
- ④ Pre increment / decrement
- ⑤ * / % → ^① arithmetic operation العمليات الحسابية
- ⑥ + -
- ⑦ > < <= >= → ^② relational operation العمليات العلائقية
- ⑧ == != → مساوية، لا مساوية
- ⑨ && → and | , &&, || → ^③ logical operation العمليات المنطقية
- ⑩ || → or
- ⑪ =

* شروط (&&) and و (||) or

x	y	x && y	x y
1	1	1	1
1	0	0	1
0	1	0	1
0	0	0	0

→ math function āzīmā, āzībā

```
① #include <cmath>
```

```
*include <iostream>
```

```
using namespace std;
```

```
int main ( )
```

```
{ int x=2, z=3, f=2;
```

```
bool y=100;
```

```
cout << !y << endl;
```

```
cout << ++x + 2 + x << endl;
```

```
cout << x + 2 * 5 + x++ << endl;
```

```
cout << !y * 5 - z++ << endl;
```

```
cout << pow(x, f) << endl; → Power → xf
```

```
cout << z << endl;
```

```
return 0; }
```

① ++x + 2 + x → 3 + 2 + 3 = 8 → dī āzīmā āzībā pre hā āzībā

② ^③x + ^①2 * 5 + ^②x++ → 3 + 10 + 3 = 16

→ āzīmā āzībā āzībā āzībā
x ||

③ !y * 5 - z++ → 0 * 5 - 3 = -3

④ pow(x, f) 4² = 16

!y → not *

x || y → āzībā āzībā

```
② bool x=1;
```

```
bool y=0;
```

```
bool z=0;
```

```
cout << !z << endl;
```

```
cout << x && y << endl;
```

```
cout << x || y << endl;
```

```
cout << y &&& z << endl;
```

1
0
1
0

* بعض الرموز التي تتبع cout ومعانيها :-

- تتبع فراغ كبير tap \Rightarrow "ا" ②
 ① "n" \Rightarrow endl .
 ③ "b" \Rightarrow يرجع الكود بشرط معرف
 ④ "r" \Rightarrow يرجع الكود بشرط بداية لسطر
 ⑤ " | " \Rightarrow " " \Rightarrow تطبع ما داخلها بين " "
 ⑥ " || " \Rightarrow " | " \Rightarrow تطبع ما داخلها بين " | "

① cout << "Jordan \n" ;	Jordan
cout << "Jordan \t first" << endl ;	Jordan first
cout << "Jordan \b \b mn" << endl ;	Jord mn
cout << "Jordan \r amm" << endl ;	amm Jordan
cout << " Jordan " << endl ;	Jordan
cout << "\" Jordan \"" << endl ;	" Jordan "

* مثال يوضح كيفية تقسيم الإدخال على عدة متغيرات :-

int x ; char c ; float z ;

- ① cin >> x >> z ; // 34.5
- ② cin >> z >> c >> x ; // 34.5 9734.52
- ③ cin >> x >> c ; // 97
- ④ cin >> c >> x ; // 97
- ⑤ cin >> z >> x ; // 34.52
- ⑥ cin >> c >> x ; // 34.52
- ⑦ cin >> x >> c ; // 34.52
- ⑧ cin >> c ; // 97
- ⑨ cin >> x >> c >> z ; // 34 22.3
- ⑩ cin >> z >> x ; // 34.252682
- ⑪ cin >> x >> c >> z ; // 34 22.3

	int <u>X</u>	char <u>C</u>	float <u>F</u>	
①	34	-	0.5	→ بيظهر باللي بيض
②	734	9	34.5	من العدد السابق
③	97	wait!	-	→ ينتظر الا ديفال اعليه ناقص
④	7	9	-	
⑤	wait!	-	34.52	
⑥	4	3	-	
⑦	34	0	-	→ 34 (0.52)
⑧	-	9	-	
⑨	34	A	22.3	
⑩	82	-	34.2526	→ Float اعز ب خانة
⑪	3422	0	0.3	

* String :-

* هي عبارة عن data type يعرف فيها متغير يخزن داخله
العديد من الحروف والصوت والاسماء

* يتم تعريفه باستخدام <code>include <string></code> ←

* يمكن الإدخال باستخدام <code>cin</code> ولكن هناك مشكلة مع
<code>string</code> ، لأنه إذا شمل الإدخال المسافة (whitespace) فهو

يقف الإدخال تلقائياً ، لذا نحن نستخدم أداة الإدخال

`getline` ← `getline (cin, string variable)`

① `string x = "ahmad";` و `y;`

`cin >> y;` // hhh hhh

`cout << x;`

hhh

② `string y;`

`getline (cin, y);` // hhh hhh

`cout << y;`

hhh hhh

* هناك بعض الـ Function التي تستخدم مع الـ string لتعلم
حجم البيانات فيها مثل
`strvar.length()`
`strvar.size()`

③ `string x = "ahmad";`

`cout << x.length() << endl;`

`cout << x.size() << endl;`

5

5

مثل ما يوضح المثال السابق فإن كلا الرالتين تعلمان على
عدد الحانات (يطبع طول النص)

* المقارنة بين المتغيرات من نوع string :-

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

Ⓐ	str1 < str2	✓	Hello	Hi
Ⓑ	str1 > "then"	x	Hello	then
Ⓒ	str3 < "An"	✓	Air	An
Ⓓ	str1 = "hello"	x	Hello	hello
Ⓔ	str4 >= "Billy"	x	Bill	Billy

* عند المقارنة بين كلمتين لا يؤثر عدد حروفها ، يؤثر إذا كان الحرف كبير أو صغير $a \Rightarrow 97$ ، $A \Rightarrow 65$

* يتم المقارنة حرف بحرف ضمن الكلمتين وكلما بعد الحرف عن حرف الـ a كان أكبر ف b أكبر من a و c أكبر من b وهكذا

Control Structure

selection
if, switch

repetition
for, while, do while

If :-

* If هي عبارة شرطية ، تختبر فيها الشرط على المتغير
* اذا كان جواب الشرط صحيح ننفذ عمل if ، اذا كان خطأ
ننفذ عمل else .

* if (condition)

```
{
  ---
  ---
}
```

} body of if

else / optional

```
{
  ---
  ---
}
```

} body of else

* لا يوجد بعد أقواس if فاصلة منقوطة

* else || اختيارية ويمكن حذفها

* اذا كانت جملة if أكثر من جملة تنفيذية بشرط وضع أقواس syntax

* اذا كانت جملة else أكثر من جملة تنفيذية ولم يتم وضع الأقواس

← يعتبر الجملة الأولى هي فقط جملة else والباقى تنفيذ بالترتيب
الأصلي .

* كما يمكن سؤال داخل شرط if على كذا يساوي كذا نضع (==) اما

إذا وضعنا واحدة فقط (هنا يعني على قيمة x = قيمة معينة أو يعني

على قيمة المتغير وحطها كذا وعادة يكتب (logical) لأنواع

القيمة وما سؤال عن الشرط

* عبارة if هي عبارة شرطية (تعمل على تنفيذ أو عدم تنفيذ)
 * إذا كان داخل أقواس if فقط ال variable تكون جواب الشرط true إذا كان اسم غير العنصر وعندها يكون مع جواب الشرط false.

```

    ① int grade = 50 ;
      if ( grade <= 50 )
      {
        grade + = 10 ;
        cout << "grade = " << grade << endl ;
        cout << "Pass" ;
      }
  
```

grade = 60
Pass

```

    ② int x = 5 ;
      if ( x < 5 )
      {
        cout << x + 1 ;
      }
      else
      {
        cout << x - 2 ;
      }
  
```

3

```

    ③ int x = 3 ;
      if ( x < 5 )
      {
        x + = 2 ;
        cout << x ;
      }
      else
      {
        cout << x + 1 << endl ;
        cout << "good" ;
      }
  
```

* إذا ما طبقنا أقواس else
 2. لتو تكون أول كلمة فقط
 والله بعد بنوع البرنامج
 الرئيسي.

5
good

```

    ④ int x = 2 ;
      if ( x < 3 )
      {
        cout << x << endl ;
        cout << x + 2 << endl ;
      }
      else
      {
        cout << x + 2 ;
        cout << "good" ;
      }
  
```

* لا يجوز الفصل بين if و else بعد فاصلة
 وهنا اعتبر أول كلمة فقط (if والباقي ؟)
 هنا الخطأ ← لا يمكن بعد وبع
 الأقواس .

→ Syntax error


```

5) int x = 2;
   if (x >= 2)
   {
       if (x > 5)
       {
           cout << x;
       }
       else
       {
           cout << "thank you";
       }
   }
   }
   }

```

thank you

```

6) int x = 4; float y = 2;
   if (x % y == 0)
   {
       cout << "OK";
   }
   else
   {
       cout << "no";
   }

```

من قواعد ال mod ان لا يكون
float

```

7) int x = 1;
   if (x = 2)
   {
       cout << x + 1;
   }
   else { cout << x; }

```

من قواعد ال mod ان لا يكون
logical error

```

8) int a = 3, b = 2;
   if (a == b++)
   {
       cout << "value of a:" << a;
   }
   else
   {
       cout << "value of b:" << b;
   }

```

من قواعد ال mod ان لا يكون
variables ال
نظير ال

```

9) int x = 1;
   cout << x + 1;
   int y = 2;
   cout << y + 1;

```

x	y
1	2
2	3

```

10) int x = 1;
    if (x)
    {
        cout << "OK";
    }
    else cout << "no";

```

من قواعد ال mod ان لا يكون
true
false

```

11) int x; cin >> x;
   if (x == 5)
       cout << static_cast <double> (5+8);
   else if (x == 2)
       cout << static_cast <double> (15/2);
   else if (x == 1)
       cout << static_cast <int> (7.8 + static_cast <double> (15/2));
   else if (x == 3)
       cout << static_cast <double> (15/2);
   else if (x != 7)
       cout << static_cast <int> (7.8 + static_cast <double> (15/2));

```

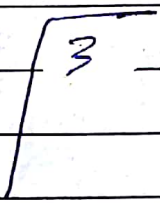
<u>cin x</u>	<u>out Print</u>
5	13.0
2	7.5
3	7.0
1	15
4	14
7	Print nothing.

12)

```

int x; x = 3;
if (x > 5);
cout << x;

```



في هذه الحالة if شرطه هو الفاشل
 المتوقعة و يعتبر باطل
 اكل من البرنامج الرئيسي

```

13) int x; x = 3;
   if (x > 5)
       cout << x;
   else

```

في هذه الحالة if هو صحيح في
 فاعتبر else شرطه غير صحيح
 بعد if

```

cout << ++x; ~> Syntax error

```


* coding ex :-

① Write a program to input 3 numbers and print the max value.

```
#include <iostream>
```

```
using namespace std;
```

```
int main ( )
```

```
{ double x, y, z;
```

```
cin >> x >> y >> z;
```

```
double max = x;
```

```
if ( y > max )
```

```
max = y;
```

```
if ( z > max )
```

```
max = z;
```

```
cout << max;
```

```
return 0; }
```

② Write a program that ask a user to enter a grade and print the result as :-

90 - 100 → A

80 - 89 → B

70 - 79 → C

50 - 69 → D

< 50 → F

```
#include <iostream>
```

```
using namespace std;
```

```
int main ( )
```

```
{ cout << "Enter your grade" << endl;
```

```
int x;
```

```
cin >> x;
```

```

if (x >= 90 && x <= 100)
cout << x << endl << "A" << endl;
if (x >= 80 && x <= 89)
cout << x << endl << "B" << endl;
if (x >= 70 && x <= 79)
cout << x << endl << "C" << endl;
if (x >= 50 && x <= 69)
cout << x << endl << "D" << endl;
if (x < 50)
cout << x << endl << "F" << endl; return 0; }
else cout << "fail";

```

③ Write a program that ask user to enter a value and print if this value even or odd.

```

#include <iostream>
using namespace std;
int main( )
{ cout << "Enter the value" << endl;
int x; cin >> x;
if (x % 2 == 0)
cout << "even" ;
else
cout << "odd" ;
}

```


هناك طريقة كتابة if بـ C++ وتسمى if statement

① condition ? C++ ; C++
sentence sentence

② cout << (condition ? some ; some) ;
thing thing

كثير من هذا النظام وجود else ، والا يخطئ syntax error
من عيوبها انها تأخذ حصة تنفيذية واحدة

grade >= 60 ? cout << "Pass" ; cout << "Fail" ;
condition if body of if else body of else

بعض الفروقات بين الفوئجين ① و ②

الاول يمكن عمل جميع انواع اكل اما الثاني فقط cout
يمكن كتابة حزين الفوئجين بحالة if العادية وليس العكس

* int x ; x = 10 ;

x >= 11 ? x++ ; cout << x++ ;

cout << x ;

	x
10	10
11	11

* Constant (const)

* تقوم const بتثبيت قيمة المتغير وعدم تغييرها عند تنفيذ البرنامج .

وغير تغيير قيمته داخل البرنامج — syntax error

const int x ; x = 4 ; —> syntax error *

Switch :-

- * switch يطلب ادخال قيمة أو تعريف قيمة من قبل ال user ثم يبحث عن ال value التي يطابقه قيمة ويمنح البرنامج من أول break .
- * إذا أتت فائدة منقطة بعد أقواس switch ← syntax error .
- * ال break يمنع البرنامج ويخرج منه قلمًا .
- * قيمة بعد ال رقم قيمة ، يقوم بتعيينها إلى ال cases دون التكرار كما في قيمة .
- * ال switch لا تأخذ إلا ال int أو char .
- * إذا كان داخل switch (x++) صحيح أنها تزيد القيمة بعد الخروج من الكمية وبقية تنفيذ ال case التي تحصل القيمة الأخيرة لها .
- * القيمة الأخيرة داخل الأقواس .

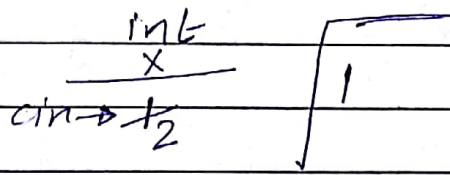
Switch (expression)

```

{ case value 1 : ----- ; break ;
  case value 2 : ----- ; break ;
  :
  case value n : ----- ; break ;
  default :
  ----- ;
}

```

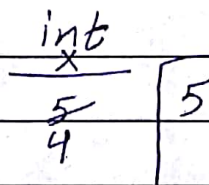
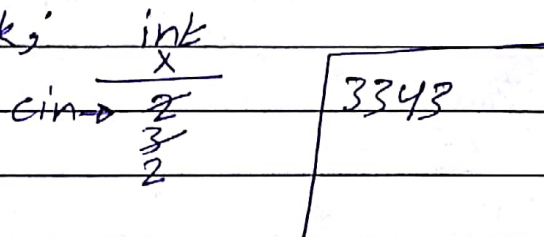
① int x=1 ; cin >> x ;
switch (x)



```

{ case 1 : cout << x++ ; break ;
  case 2 : cout << ++x ;
  case 3 : cout << x ;
  default :
  cout << x-- ; }

```



② bool x = true;

switch (x)

{ case 1: cout << x++; break;

case 2: cout << ++x; break;

case true: cout << ++x; break;

case 97: cout << x; break;

case 'a': cout << x--; break;

}

→ syntax error

{ 1 == true

97 == 'a'

لا يجوز ان يكون أكثر من case في القيمة.

③ int x = 1;

switch (x++)

{ case 1: cout << x; break;

case 2: cout << x+5;

}

$\frac{x}{2}$ 2

* في خروج التفاضل من case switch يزيد بمقدار واحد ولا يزداد case الى case التالية قبل الزيادة (x++)

④ int x = 1;

switch(x) { } → سبب الفشل . خطأ

{ case 1: cout << x++;

case 2: cout << ++x;

→ syntax error

الفروق بين البراء سلاش والقسمه

✓ "\n" ✓

✓ 5/2 ✗

for :-

* For على تكرار تنفيذ الكود التي تحتوي على لها عدد معين.

```
for (int i=1 ; i<=2 ; i++)
```

```
{
  --
}
```

ترتيب الخطوات عند التنفيذ ←

```
cout << --- ;
```

```
① for (int i=0 ; i<=2 ; i++)
```

```
{
  cout << i << endl ;
  cout << "hello" << endl ;
}
cout << "ok" ;
```

i	0
0	hello
1	1
2	hello
3	hello
	2
	hello
	ok

```
② int i=0 ;
```

```
for ( ; i<=3 ; i++)
  cout << i << endl ;
  cout << i+2 ;
```

0	* يمكن كتابة امر اجزاء
1	* For فوقها
2	* عند عدم وضع i تكون القيمة
3	الأولى هي For
6	

* عند الخرج من for فإن قيمة المتغير تزيد على القيمة الـ default
بمقدار مرة واحدة لتنفيذ الكود التي تلي For

```
③ for (int i=0 ; ; i++)
```

```
  cout << i ;
  cout << "ok" ;
```

0	لا يوجد
1	لا يتوقف
2	تكرار
...	
...	

infinite Loop

④ For (int i=0; i<3;)
 { cout << i; }

0
0
}

infinite loop

لأنه الرقم لا
 يزداد فيبتكر
 ال loop دون توقف

⑤ For (int i=0; i<3;)
 { i++;
 cout << i; }

1
2
3

* اختلافه ان الزيادة في
 قيمة i قبل الطباعة .

⑥ For (int x=1; x<20; x*=5)
 cout << x;

x
1
5
25

1
5

coding ex :-

① Write a program that print the numbers between (1-10) and print if it even or odd.

#include <iostream>

using namespace std;

int main ()

{ For (int x=1; x<=10; x++)

if (x%2 == 0)

cout << x << "even" << endl;

else

cout << x << "odd" << endl;

return 0; }

② write a program to enter 5 numbers and print if it positive or negative.

```

#include <iostream>
using namespace std;
int main ( )
{
    int y;
    for (int x=1; x<=5; x++)
    {
        cin >> y;
        if (x>=0)
            cout << x << "positive" << endl;
        else
            cout << x << "negative" << endl;
    }
    return 0;
}

```

③ write a program its output is →

```

****
***
**
*

```

```

for (int i=4; i>=1; i--)
{
    for (int x=1, x<=i; x++)
        cout << "*";
    cout << endl;
}

```

④ write a program that find 5! ^{مربع} _{الرقم}

```

int x, y=1;
cin >> x;
for (i=1; i<=x; i++)
    y = y * i;
cout << y;

```

⑤ write a program that find sum of numbers (1-10)

```

int x; int sum=0; cin >> x;
for (int i=1; i<=x; i++)
    sum = sum + i;
cout << sum;

```


* عند السؤال عن طباعة عدة صفوف وأعمدة كتكرار طباعة 6 نجوم مرتباً على 3 صفوف متتالية تتبع الآتي -
 - نستخدم `rested for` وهو موجود في `for` داخل عدة `for` أخرى،
 حيث تمثل الجملة الأولى "العمود" عند الصفوف، وتمثل الثانية
 عند الأعمدة

إذا كان عند الأعمدة ثابتة في كل الصفوف يفرض ال `condition`
 رقماً ثابتاً (لا `for` الثانية)
 إذا كان عند الأعمدة غير ثابتة في جميع الصفوف يفرض ال `condition`
 بالنسبة للتغير الأول (لا `for` الثانية)
 - ارجع للتاليين السابقين للتفصيل.

for with char :-

تذكير ببعض قواعد char :-

```
* char x = 'a'; cout << x;
* char x = 'a'; cout << x;
* char x = '97'; cout << x;
* char x; cin >> x; // 97
```

```
① for (char x = 'a'; x <= 101; x++)
    cout << x << "\t" << int(x) << endl;
```

a	97
b	98
c	99
D	100
E	101

x
 a → 97
 b → 98

```
② char x = 'a';
```

```
x++;
cout << x;
cout << x+1;
```

x
a
b

b
99

إذا قمنا بزيادة متغيرنا
 على char ولم نغيرها
 نتعامل مع قيمتها

★ For (int i=1; i<=3; i++)

if (i%2 == 0)

break;

cout << i;

}

cout << "ok";

break داخل

if

for

(for داخل if)

★ For (int x=1; x<10; x++)

if (x%2 == 0)

continue; → C++ keyword.

cout << x << endl;

}

↳ برنامج

for لا تعمل قريسة ارجع لل for

تأنيو الـ continue لا تقبل ان يكون على طرف if لطباعة الارقام الفردية.

★ For (char e = 'a'; e <= 'd'; e++)

if (e == 'c')

continue

cout << e << endl; }

a
b
d

While :-

while (expression)

{

// body of while

}

* For while loop the expression must be true before the execution of the body.
 * while (do) loop (while-do) executes the body first and then the expression.
 * while (do) loop (while-do) if the expression is false, it will give a syntax error.

* If the condition is true, the body will execute.
 * while (do) loop (while-do) if the expression is false, it will give a syntax error.
 * If the condition is true, the body will execute.
 * while (do) loop (while-do) if the expression is false, it will give a syntax error.

① int x=5;	5	x
while (x <= 7)	6	5
{ cout << x;	7	6
x++;	8	7

② int x=5;	x	4
while (x-- >= 1)	5	OK
{ cout << x << endl;	4	3
cout << "ok" << endl; }	3	OK
cout << "done" << endl;	2	2
	1	OK
	0	1
	OK	OK
	0	0
	OK	OK
	done	done

rand :-

32767, 0...
→ rand.max

#include <C std lib>

فيسين ا

```
* for (int i=0; i<10; i++)
```

```
cout << rand ( );
```

10
5
2
1
1

Range ال...
تقليل من الاعتدالات :-

off set + rand () % scaling factor .

→ ...

أول... كذا...

* min = off set

* max = scaling factor + min - 1

...
== scaling factor

فيسين ا

```
① For (int i=2; i<=5; i++)
```

```
cout << 1 + rand ( ) % 6 ;
```

- ① 153426
- ② 1153
- ③ 115
- ④ 1761

[1,6] ← Range ال

min = offset = 1

max = scaling factor + min - 1 = 6 + 1 - 1 = 6

اعد قسيه

②... ارقام 4... اعد قسيه...

```

② int x=1;
   while ( x < 3 )
   { cout << ( 2 + rand ( ) % 6 );
     x++;
   }

```

- ① 78 ② 77 ③ 473 ④ 783

min = 2
max = 6 + 2 - 1 = 7 Range [2, 7] → while دورتین

الجواب ②

لا تحسبن بل مجرد تقرأ أنت
الله
لن تبلغ بل مجرد صحتي تلعبت
العبيرا

مع تقنيات بالتوفيق والنجاح
مرور عاونه
هندسة حاسوب

تلخيص مراجعة

عادة الفايثال

لبرمجة الحاسوب

C++

(المهندسة)

اعداد الطالبة :- عروه حلوه

دعتر المهندس الاد ابو سرحان

Function :-

* هو data type يستخدم خارج الـ main الاساسي ، يعمل على ادخال قيم والقيام بالعمليات الحسابية والمنطقية على القيم وارجاعها لكي الـ main الاساسي .

* void تعني return nothing

* يسمى استخدام الـ Fun داخل الـ main بالاستدعاء (calling)

* يسمى ادخال الارقام ، الى الـ Fun (القيم ، المتغيرة) بـ Parameter passing

* الـ data type ، المستخدمة في الـ Fun تسمى بـ return data type ما عدا الـ void

* اذا كان return data type يجب وضع return

* اذا وضع return مع void (syntax error)

* لا يجوز طباعة قيمة الـ Fun نوعه void ، طباعته في الـ main وليس محل الطباعة فيه ركز ؛ لاننا اعلمنا بالوقتية في الـ main لاننا ما برجع اشي مما نريد نطبعو .

* اسماء المتغيرات في الـ main تختلف عن اسماء المتغيرات في الـ Fun والمتغيرات المعروفة في الـ main لا تعتبر معروفة في الـ Fun ، كل واحد ل حال .

* يعتمد الـ Fun على ترتيب المتغيرات وليس على اسمها ، هاد في جملة تعريف الـ Fun هاي اللي يتكون اول سطر في الـ Fun .

* الـ proto type وهو تعريف الـ Fun قبل الـ main عند وضع الـ Fun تحت الـ main (عشان ما يعنى syntax error) عادي عادي في امثلة بتوضع كقول مخلص .

* الفكرة انو اذا خطينا الـ Fun بعد الـ main يرج يعنى syntax لاننا كتبنا اننا جوا الـ main اسمو والبرامج ما يعرف انك كاتبو تحت فلازم نخط جملة قبل نكتب فيها انو في الـ Fun تحت الـ main فينخط الـ proto type .

* يتم عمل casting لقيمة الـ return حسب نوع الـ return data type (اللي هي نوع الـ data type اول جملة قبل اسم الـ Fun) .

* عندما يتم عمل الـ return فانها تعود الى عمليات الـ call ولا تغير من القيم الرئيسية للمتغيرات في الـ main الاساسي

Pass/call by value :-

- * بمعنى انك ترسل قيمة المتغير الموجودة داخل الـ main الى الـ Function المتغير الموجود في الـ Function
- * اذا كان الـ fun يأخذ 2 Parameter وتم ارسال واحدة فقط syntax error ←

Some Notes :-

- * اذا ما اعطيت المتغير أو ما بين أقواس الـ fun قيمة لا في الـ Prototype ولا في الـ call syntax error ←
- * اذا كان احد المتغيرات الـ local ناقصه فإن الـ global var لا محل محله ، يستخدم فقط بالاستدعاء (احد متغيرات الـ fun)
- * ما بصير الـ static داخل أقواس الـ fun
- * طول ما أنا ما اعطيت للمتغير قيمة ← syntax error
- * عند اعطاء المتغيرات الخاصة بالـ fun قيم داخل الـ Prototype فإنها يعود إليها عند وجود ناقصه عن عدد المتغيرات في الـ call
- * يأخذ القيم من الـ Prototype بترتيبهم فيه (اذا كان لبقه في الأخير يأخذ الأخير وهكذا)
- * لا يمكن ترك فراغات في الـ call الـ call
- غلط ← ex :- fun (, , 2) ;
- * نحن نعتبر التعريف المسبق (Proto) للـ fun فإننا يجب أن يعرف نفس عدد الـ var
- * اذا وضعنا داخل الـ call عدد متغيرات/قيم أكثر من المتغيرات داخل الـ fun syntax error ←

* اجزاء ال Function

```

return data type
int addition ( int a , int b )
{
    int r; // Function's name
    r = a + b;
    return r;
}

void main ( )
{
    int z;
    z = addition ( 5 , 3 ); // Parameter Passing
    cout << z; // calling sentence.
}
    
```

8

* هسا نذكر شئو هسا ر عشان تطلع ال 8 ، اول اسئ بروج على ال main الاساسي بعدما اتأكد انو ما فيه ولا خطأ كتابه من تعريف ال Fun يعني اول جملة صح فيها ال return data type واسم ال Fun وكل اسئ تمام ، هسا بترك الاسئ جوا ال Fun و بروج على ال main زي ما حكيينا ، بقتبدأ اول اسئ عرفت var ، بعدين حكا انو يساوي اسئ ، عاد الاسئ فسطو calling أو استدعاء بي عمل انو يفعل ال Fun يعني وديين القيم اللئ جوا الأقسام على ال Fun واستعمل عليها ، هسا بختبو (قيم بنقش) اسم ال Fun هسا بطلع على ال Fun ، ال Fun بعرفه فيه القيم من أول و جديد لانو بقدره - الحالو الو متغير اتو الحالو ف هون فرقت a و b و جمعهم و خصيتهم جوا r بعدين رجعت r ب return ، هسا اي اسئ جوا ال return يرجع مكان جملة ال calling هسا يرجع على ال main و بجعل باقى البرنامج .

Ex :-

```

int add ( float a , float b )
{
    float i;
    i = a + b; // 5.2 + 7.3 = 12.5
    return i;
}

void main ( )
{
    int z = add ( 5.2 , 7.3 );
    cout << z;
}
    
```

* برجع القيمة حسب ال casting ← return data type

12

3

② void add (float a, float b)

{ float r;

r = a + b;
cout << r; }

void main ()
{ add (1.5, 3.2); }

* استنوع void لـ r

* كذا void ما ينفج اعل cout
calling sentence لانوا اصله
ما رجلا قيمة .

4.7

③ void add (float a, float b)

{ float r;
r = a + b;
cout << r; }

void main ()
{ int x, y;

cin >> x >> y;
add (x, y); }

* ما د عشان اعل اذغال
قيمة x > y / a > b

④ char What (int x, float y, int z)

{ cout << y << endl;
return z; }

void main ()
{ cout << What (3.2, 5, 'C'); }

5.0
C

⑤ void print ()
{ cout << "I am function" << endl; }

void main ()
{ print (); }

I am function

* اذا كانه cout << print ()
void syntax

⑥ int sub (int x, int y)

{ return x - y; }

void main ()
{ cout << sub (5, 6) << endl;
int z = sub (10, 4);
int x = 3, y = 1;
cout << sub (x, y) << endl;
cout << z; }

-1
2
6

Pass by value

4

7) `int add (int x , int z)` ; Proto type

هون لانو طينا ال fun
 كت ال main فاول ما ربح
 يوصل ال add بربح يعني syntax
 لانو غير معرفت ، عشان هين
 بنط . حلة ، تعريف ، بسبب
 (Proto type) بتختلف انا
آخرها فاصلة منقوطة .

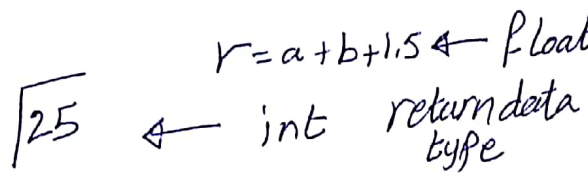
```
void main ( )
{ cout << add ( 20 , 4 ) << endl ; }
int add ( int a , int b )
{ int r ;
  r = a + b ;
  return r ; }
```



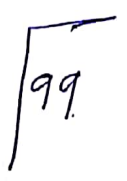
* اذا سأل عن proto type :-

- 1) `int add (int , int) ;`
- 2) `int add (int a , int b) ;`
- 3) `int add (int x , int z) ;`

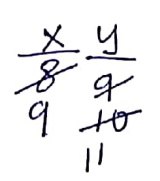
* ما بتفرق شو طينا بجد int
 * فيه امثال السابقة :- اذا كانت float
 ← بتطلع العنبر 25.5 ولانو
 ← `int return data type`



8) `int fun (int x , int y)`
`{ return 'c' ; }`
`int main ()`
`{ cout << fun (5 , 6) << endl ;`
`return 0 ; }`



9) `int end (int) ;`
`void first (int) ;`
`void main ()`
`{ first (8) ; }`
`void first (int x)`
`{ if (x < 10) * ++ ;`
`cout << end (x) << endl ;`
`int end (int y)`
`{ switch (y)`
`{ case 9 : ++ y ;`
`case 8 : ++ y ; }`
`return y ; }`



5

سؤال حلوه شوي حاول حلوا الحالك من 10
الجواب صح يكون مني لصفحة التالية
بس حاول عني الحالك .

```
char foo (int ) ;  
int bar ( int , int ) ;  
void main ( )  
{ int x ;  
  for ( x = 4 ; x != 6 ; x = ( x + 3 ) % 11 )  
    cout << foo ( bar ( x , 6 ) ) ; }  
int bar ( int a , int b )  
{ if ( a > b )  
  return b ;  
  else  
  return a ; }  
char foo ( int n )  
{ switch ( n )  
{ case 0 : return 'B' ; break ;  
  case 1 : return 'A' ; break ;  
  case 2 : return 'D' ; break ;  
  case 3 : return '!' ; break ;  
  case 4 : return 'G' ; break ;  
  case 5 : return 'J' ; break ;  
  case 6 : return 'O' ; break ;  
  default : return '*' ; break ;  
}  
}
```

سؤال كثير بس جود شويه
تركيز .

GOODJOB!

← جواب 10 *

رسالة حلوة بنفع الدراسة . نحمدك

```

(11) void While ( int x, int y) → return ما في return هون
    { x++; cout << x+y; }
void main ( )
{ int x=5;
  While ( x,x);
} capitals → Pass by value

```

* الفاصلة المنقوطة عند while يعني قبل لف في الشرط عن ريسر غلط . معلومة قدما

```

(12) void While ( int x, int y)
    { cout << x+y << endl; }
void main ( )
{ int x=5;
  while (x--);
  cout << x << endl;
} → small

```

* لو كانت capitals & while مع يكون syntax لانو الـ Pun بيوفر Parameter

```

(13) char Ok ( int ff )
    { if ( ff == 99 )
      { ff += 5;
        return ff + 1; }
      else
        return ff; }
void main ( )
{ int x = 97;
  int y = x++ + 1;
  cout << Ok ( x ) << endl;
}

```

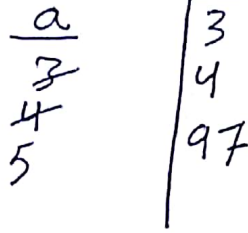
7

14) int For (int a)

```
{ for ( ; a < 5 ; a++)
  cout << a << endl;
  return a; }
```

```
void main ( )
{ int y = 5;
  cout << For(3) << endl; }
```

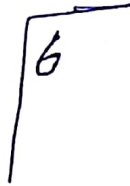
نتیجہ انویٹ *
int



15) float S (int a)

```
{ return a; }
```

```
void main ( )
{ int x = 5;
  cout << S(++x) << endl;
}
```



16) float S (int a)

```
{ return a; }
```

```
void main ( )
{ int x = 5;
  cout << S(x++) << endl; }
```



17) float S (int a)

```
{ return ++a; }
```

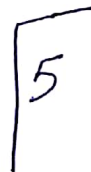
```
void main ( )
{ int x = 5; cout << S(x) << endl; }
```



18) float S (int a)

```
{ return a++; }
```

```
void main ( )
{ int x = 5;
  cout << S(x) << endl; }
```



8

19) int integer ()

```
{ return 10 ; }
```

char character ()

```
{ return '7' ; }
```

void main ()

```
{ int x = integer ( ) - 5 ;
```

```
  char a = character ( ) ;
```

```
  cout << a << endl ;
```

```
  if ( a == '7' )
```

```
  cout << x ;
```

```
  }
```

7
5

* ننتبه لترتيب
cout فيه اكل

Note :-

① void Default (int a=1 , int b) ;

خاصة (syntax) لازم يجب ان يكون، ليعمل على اليمين، لازم
فيه حال ارسال قيمة واحدة فقط (3) Default مثلا سوف تعمل لـ a
والـ var b لن يكون له قيمة ← syntax

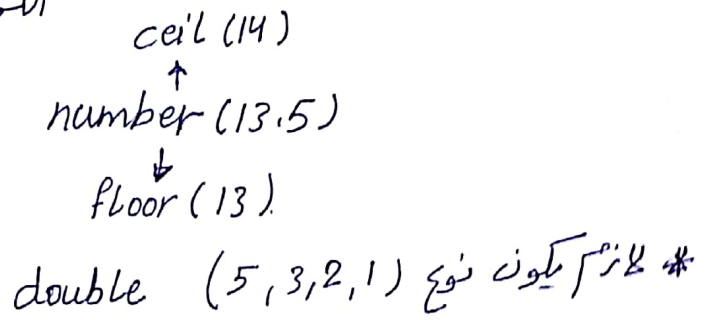
② void Default (int a , int b , int c=5) ;

ستكون صحيحة اذا ارسل 3 قيم او قيمتين لازم ارسال قيمة واحدة
الكل، اما اذا ارسل قيمة واحدة فقط ← syntax

Pre Define Function :-

- ① sqrt \Rightarrow الجذر التربيعي .
- ② fabs \Rightarrow float absolute القيمة المطلقة
- ③ pow \Rightarrow Power / الأس / القوة
- ④ fmod \Rightarrow لاستخدام باقي القسمة اذا كان احد ال float var
- ⑤ $\log_{10} () \Rightarrow$ اللوغاريتم

⑥ floor and ceil \Rightarrow



ceil = floor = number \leftarrow اذا كان الرقم عدد صحيح

floor :- Largest integer not greater than X

ceil :- smallest integer not less than X

* اذا تم تخزين هذه ال Functions في int casting ال casting
 اجريت مباشرة \leftarrow تأخذ بشكل افتراضي double

* عند وضع قيمة سالبة داخل sqrt Logical error

size of (a tan (-1)) *

هذا خطأ ال size \leftarrow size = int size \leftarrow لا يتم لقيمة ال tan

* عند اطلاق عبارة عن Functions معرفة بشكل تلقائي داخل البرنامج \leftarrow * include < cmath >

```

① #include <iostream.h>
#include <cmath>
void main ( )
{ double x = sqrt (36);
  cout << x << endl;
  double y = log10 (1000);
  cout << y << endl;
  cout << fabs (-2.3) + pow (2, 2);
}

```

```

6
3
6.3

```

```

② double x;
x = fmod (36.4, 2);
cout << x << endl;

```

```

0.4
13 13 13
14 14 13

```

```

cout << floor (13.5) << "\t" << floor (13.2) << "\t" << floor (13) << "\n";
cout << ceil (13.5) << "\t" << ceil (13.2) << "\t" << ceil (13) << endl;

```

```

③ int x = fabs (36.4);
int y = pow (2.5, 2);
cout << x << "\t" << y << "\n";
cout << fabs (36.4) << "\t" << pow (2.5, 2) << endl;

```

```

36 6
36.4 6.25

```

```

④ int y = ceil (-2.5);
cout << y;
int x = sqrt (-25);
cout << x << endl;

```

```

-2
Logical error

```

Logical

```

⑤ double i = pow (fabs (-10), 2) + pow (10, 1) + ceil (0.3);
double y = sqrt (25 + pow (5, floor (0.3)));
cout << i << "\n" << y;

```

```

111
6

```


⑥ int x=2;

switch (size of (atan(-1)))

{ case 2: cout << pow(x, 2) << endl; break; }

case 4: cout << x << endl; break; }

default:

cout << pow(x, 3) << endl; }

2.

لما تَكْسِر قلبك قَل يا رب يا جبار م
فإذا بالأكسر - بحبر قادر على ما لا يقدر ♥

* الفكرة باستخدام نفس اسم الـ var داخل نفس البرنامج Scope :-

* هو تعريف للشيء، المعرف فيها الـ variable

* الفتق التي يمكن استخدام الـ variable دون أن يعطي syntax error

* الوقت الذي يبقى فيه تعرف المتغير عن الذاكرة (memory time)

* عند تعريف متغير داخل الـ main تكون main fun scope

* عند تعريف متغير داخل الـ fun تكون fun scope

* داخل الـ Scope الواحد :-

- يبدأ وينتهي بـ { }

- يمكن وضع أقواس داخلية ، عند وضعها تعتبر scope منفصل .

- تنقسم الـ scope الكبيرة بـ outer block scope و الـ scope الداخلية inner block scope

- تعرف المتغير في inner scope يمكن الاستغناء عنه ، يستخدم قيمة المتغير في outer block scope

- تعتبر أقواس if , for , while inner scope

Global Variable :-

هو variable يتم تعريفه خارج كل من main , fun . يتم استخدامه بشكل تلقائي إذا لم يتم إعطاء قيمة للمتغير داخل الـ scope في البرنامج ويحل أفضلية استخدامه داخل البرنامج بـ (::) .

Local Variable :-

هو variable يتم تعريفه داخل الـ scope ، يستخدم داخله فقط .

* إذا عرفت global var يتم داخل الـ main ذكر الـ ~~global~~ int z = 5 ;

main
z = 3 \Rightarrow تعتبر تغيير لقيمة الـ global var أما إذا وضع int

يعتبر outer block ولا يأتى الـ global var

① `int out (int (x))` → function scope

`{ return x * 5; }`

`void main ()`

`{ int x = -5; }` → main function scope

`cout << out (x); }`

② `void main ()`

`{ int z = -5;`

`cout << z << "\n"; }` → outer block scope

`int z = 5;`

`cout << z << "\n"; }` → inner block scope

`cout << z << endl;`

-5
5
-5

③ `void main ()`

`int z = 5;`

`{ cout << z;`

`{ cout << z;`

`{`

`cout << z;`

`}`

555

④ `int z = 5;`

`void main ()`

`{ cout << z << endl;`

`{ cout << z << endl;`

`{ int z = 3;` → local variable

`cout << z;`

`}`

5
5
3

```

5) int z=5;
   void main

```

```

{
  { z=3; cout<<z<<endl; }
}
cout<<z<<endl;
}

```

* اذا كان قبل z int
 ← z يعتبر
 Local var يعتبرها

```

6) int fun(int x)
{ cout<<x;
  return x; }

```

→ syntax error

لانه تم تعريفه داخل ال fun
 وليس global فتلا فيعتبر
 استخدامه دون تعريف

```

void main()
{ x=3;
  cout<<fun(x);
}

```

```

7) void main ( )
{ int x=5;
  for (int i=3; i<4; i++)
  { int x=7;
    cout<<x; }
}

```

7

* اما اذا جلتنا للاول
 تبعات for ربع يعتبر به
 اوله لـ for ربع
 ← 5 فيه
 int الاول

```

8) void fun ( void )
{ }
void main ( )
{ fun ( );
}

```

↳ no parameter


```

⑨ int x=1;
void fun1(int);
void main()
{ cout << x << endl; }
fun1(5);
}
void fun1(int x)
{ cout << x << endl; }
}

```

```

┌
│ 1
│ 5
└

```

جلتوا عاي من جلس عاي فركبتيه
 تواسي كُفلاً عات عصفوره ♡♡

Referance :-

- * هو تعريف المتبايع متغير لتغير آخر .
- * بأشتر على المتغير وكل ما تتغير قيمة الأول يتغير قيمة الثاني
- * يعني يربط متغيرين مع بعض اي اتمني بصير على واحد بصير على الثاني
- * عشان نربط متغير بمتغير آخر نفتح توكه اشارة and (&)
- * اللي جنبو الاشارة هو يتنج / يأخذ قيمة المتغير اللي بعد ال =
- * حالات ال syntax error في ال Referance :-

① `int &y;` لانك بتعطي للبرنامج عرقة y واحجزها مكان في ال memory ورجعها ل... لشيء؟ فغشان عليك غلط

② `int &y=5;` ما بقدر اتبع متغير مع رقم / قيمة ← `int y=5;`

③ `int x=3;` `char &y=x;` يجب ان يكون ال نفس ال datatype

④ `int &y=x+1;` لا يسير الا على variable

* بين بصير تفرقت 'int x' وبتعطيها قيمة لا حقاً في البرنامج
`int &y=x;`

* `int &x+=3;` هون صح لانني خزنت لقيمة في المتغير وبعد عرضها بعض قيمة

```

① int x = 10;
   int &y = x;
   cout << x << endl;
   cout << y << endl;

```

$\frac{x}{10} \quad \frac{y}{10}$

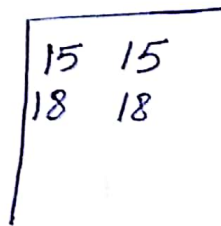


```

② int x = 10;
   int &y = x;
   x = x + 5;
   cout << x << y << endl;
   y = y + 3;
   cout << x << y << endl;

```

$\frac{x}{10} \quad \frac{y}{10}$
 $\frac{15}{15} \quad \frac{15}{15}$
 $\frac{18}{18} \quad \frac{18}{18}$

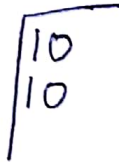


```

③ void by_ref (int &x)
   { x * = 2; }
   int by_value (int x)
   { x * = 2;
     return x; }
   void main ( )

```

* ما تنسى انوال Fun بيوجد
 اسماء كثير وشروطه زي شروط
 كتابة اسم ال var .



```

   { int y = 5;
     int z = by_value (y);
     cout << z << endl;
     by_ref (y);
     cout << y << endl;
   }

```

$\frac{x}{5} \quad \frac{y}{5}$
 $\frac{10}{10} \quad \frac{10}{10}$
 $\frac{z}{10} \quad \frac{y}{5}$
 $\frac{10}{10} \quad \frac{10}{10}$

```

④ void change (int &var) → int &x += 3 ← هاي هون *

```

```

   { var += 3; }
   void main ( )
   { int x = 5;
     change (x += 3);
     cout << x; }

```

→ x = 5 + 3 = 8

$\frac{x}{8} \quad \frac{var}{8}$
 $\frac{11}{11} \quad \frac{11}{11}$



```

⑤ void ceil_fun (double &y)

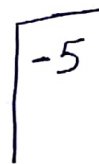
```

```

   { y = ceil (y); }
   void main ( )
   { double x = -5.9;
     ceil_fun (x);
     cout << x; }

```

$\frac{x}{-5.9} \quad \frac{y}{-5.9}$
 $\frac{-5}{-5} \quad \frac{-5}{-5}$



Storage Class :- memory داخل الـ memory

Storage class → automatic (auto, register)
→ Static (extern, static, mutable)

* Static :-

- يعطى المتغير قيمة غير تلقائياً
- اذا عرفت متغير static بدون قيمة واستخدمته لا يعطى error لان قيمة default = 0
- الـ global variable ← static by default
- Static ← تعنى بقا قيمة المتغير داخل الـ memory دائماً و يستخدم داخل كامل البرنامج

* Auto :-

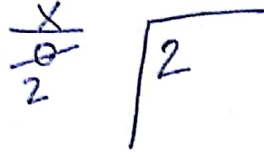
- هو المتغير العادي الذي استخدمناه من قبل
- عندما يكون المتغير من نوع auto تكون قيمته محصورة داخل block معينة
- الـ Local variable ← auto by default

* يمكن تغيير نوع الـ storage بذكر النوع قبل الـ data type
ex → static int x = 2

```

① static int x;
   x += 2;
   cout << x;

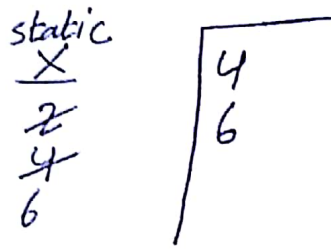
```



```

② void twice ( )
   { static int x = 2;
     x += 2;
     cout << x << endl; }
void main ( )
   { twice ( );
     twice ( );
   }

```



* ثابتة في الـ البرنامج

* الـ static يبقى في الـ memory لكل الـ برنامج
 * الـ auto يبقى في الـ block، الـ الـ الـ الـ الـ الـ

```

③ int var = 7;
   void unary ( int var )
   { var++;
     cout << var << endl; }
void main ( )
   { int var = 3;
     unary ( 5 );
     cout << :: var << endl;
   }

```



* الـ الـ الـ الـ الـ
 * global variable الـ

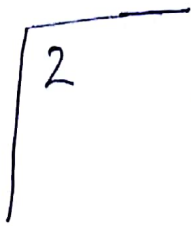
Go hard ...
 or go home .

بعض الملاحظات على تعريف الـ function :-

```

1 void Default( int x=2);
  void main ( )
  { Default ( ); }
  void Default(int a)
  { cout << a << endl; }
    
```

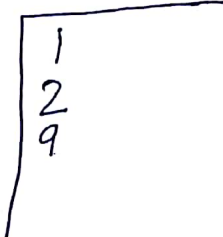
* عند اعطاء المتغيرات خاصة بالـ fun قيم داخل الـ prototype
 * إذا لم يوجد لها عند وجود فقط من قيم المتغيرات في الـ calling.



* syntax إذا ما كان في (int x=2)

```

2 void Default (int a=1, int b=2, int c=9);
  int a=1;
  void main ( )
  { Default ( ); }
  void Default (int a, int b, int c)
  { cout << a << endl;
    cout << b << endl;
    cout << c; }
    
```



* إذا كان واحد من var ناقصه ما لا يحمل مكانه الـ global ← ليس يعوضه الى جوار الـ scopes ما دخلو بالـ prototype.

* ما يسير احد static داخل اقواس الـ fun

* قول ما انا ما اعطيت الـ variable قيمة ربحه syntax error.

```

3 void Default (int a=1, int b=2, int c=9);
  void main ( )
  { Default (3);
    Default (3,3);
    Default (3,3,3); }
    
```

54	3	2	9
81	3	3	9
27	3	3	3

```

void Default (int a, int b, int c)
{ cout << a * b * c << endl; }
    
```

* إذا اعطى قيم داخل الـ main وكانت ناقصة يفرضها على الترتيب من اول fun ياخذ القيم المفروضة من الـ main ويكمل النقص من اول fun

* إذا وضع داخل دالة `call` عدد متغيرات / قيم المتغيرات أكثر من المتغيرات داخل الـ `Fun` ← `syntax error`.

④ `void Fun (int a=3, int b=4, int c=6);`
`void main ()`

`{ Fun (4,4,4,4); }`

`void Fun (int a, int b, int c) → syntax error .`

`{ cout << a+b+c ; }`

* `void Default (int a=1 , int b);`

- إذا أعطى عدد أقل من المتغيرات ← `syntax` ، لأن يراعى الترتيب فيجعل القيمة المخطأ داخل الـ `calling` للمتغير الأول ولا يكون هناك قيمة للمتغير الثاني ← `syntax`

* `void Default (int a, int b, int c=5);`

- ستكون صحيحة إذا أرسل 3 قيم أو قيمتين لأن من قيمة المتغير الأخير ، أما إذا أرسل قيمة واحدة فقط ← `syntax`.

وليس الـ `العبارة` بين `سبقت` ...
ولكن الـ `العبارة` بين `صوت` ..

Function Overloading :-

* لاستخدام fun نفس الاسم أكثر من مرة .

* يسمح تكرار الـ fun اذا عُدت الـ parameter داخل الـ fun .
فأخذ الأنسب تبعاً للقيم المعطاة في الـ calling sentence .

* لا يكون الاختلاف :-

① عدد المتغيرات ② نوع الـ data type ③ ترتيبهم .

* الترتيب مختلف لا يشمل كون الـ data type متطابقاً ← syntax error

* لا يهم اسم الـ variable مهم اختلاف الـ data type

* اذا كان الـ data type (int و double) نختار الأنسب للارقام في الـ call

- int ← int fun double ← double fun

- اذا كان هناك واحدة double على الأقل ← double fun

* اذا كانت المقارنة بين double و float ← double fun

لان الـ float المستوي by default ← double

* $\sqrt{8}$ → cout << size of (1.2)

* لا يمكن تعريف fun فارغ مع fun - كل قيم بنفس الاسم

← syntax error

* لا يمكن ذكرها بجملة call أيضاً

cout << fun () ;

* ويكون في أكثر من fun - يكون اعداد مختلفة من الـ variables

```

① int fun1( int x , int y , int z )
   double fun1( double x , double y , double z )
   void main ( )
   { cout << fun1(2,3,1) << endl;
     cout << fun1( 2.5, 3.8, 1.4) << endl;
     cout << fun1( 2.5 , 3, 1) << endl; }
   double fun1 ( double x , double y , double z )
   { return x + y + z ; }
   int fun1 ( int x , int y , int z )
   { return x + y + z ; }

```

6
7.7
6.5

```

② int fun1 ( float x , int y , int z )
   { return x * y * z ; }
   double fun1 ( double x , double y , double z )
   { return x + y + z ; }
   void main ( )
   { cout << fun1(1.2, 1.5, 1.6) ; }

```

3.2

* اذا كان بدل ال double ← int ، فلياً ، ال float .

```

③ int fun ( int x , int y )
   { return x + y ; }
   int fun ( int x )
   { return 5 * x ; }
   void main ( )
   { cout << fun(1) ;
     cout << fun(1, 2) ;
     cout << fun ( ) ; → syntax error
   }


```

* لا يمكن تعريف (fun) خارج مع (fun) -
عمل أرقام


```
(4) int funz (int x, int y)
    { return x+y; }
int fun ( )
    { int r=1;
      return r; }
void main ( )
    { cout << fun ( ) << endl;
      cout << fun(4,5) << endl;
    }
```

```
1
9
```

Don't forget to
Stop and smell the

Roses! 

Array :-

* هي طريقة لتخزين المتغيرات (variable) أكثر من قيمة في نفس المكان (مجموعة).

* ما يوجد داخل الإشارة الـ array [] هو عدد عناصر / قيم / قيمتين داخله ، أما رقم الخانة فهو (رقم الخانة من الـ array - 1)

int x[3];
 ↳ no. of elements → n
 index 0 → n-1

* يجب ان يكون int else casting (اذا كانت data type) (int)

* لا يمكن طباعة الـ array بجملة cout لا يا مجموعة عددية
 يجب كتابة رقم الـ index ← cout << x[0];

* يتم عمل cin للـ array باستخدام الـ for

* حالات يتم الـ Array :-

① int x[] = { 1, 2, 3 }; →

1	2	3
---	---	---

 -

② int x[3] = { 1 }; →

1	0	0
---	---	---

 -
 يفرغ عدد من عدد القيم

يعني باقي القيم / الخانات الفارغة بأصغر

③ int x[]; cin → (, ,) → syntax error -

④ int x = { 1, 2, 3 }; → syntax → size error -

* Array 2 dimensions :-

int x [] [] ;
 عدد الأعمدة عدد الصفوف

int x[] = { (1, 2), (5, 3) };

1	2
5	3

 * إذا تم الإدخال على شكل المصفوفة

* إذا تم إدخالها بشكل يتم ← يعني الـ array كامل يتم ينقل ، أي صف جديد ← إذا كان هناك نقص في القيم يكمل بأصغر

* إذا جلبت طباعة index أكبر من عدد الـ indexes يعني يطبع rubbish

Logical error ←

① double x[5];

```
for (int i=0; i<=4; i++)
{ x[i] = i*2; }
```

index	x
0	0
1	2
2	4
3	6
4	8

② int x[3] = {1, 5, 3};

```
cout << x[2];
cout << x[0];
```

index	x
0	1
1	5
2	3

3	1
---	---

③ int x[5][2] = { (1,2), (1,4), (2,0) };

```
cout << x[2][0];
```

	x	
	0	1
0	1	2
1	1	4
2	2	0
3	0	0
4	0	0

2

④ int x[2][3] = { 1, 2, 3, 4 };

بشكل أسفار

1	2	3
4	0	0

⑤ int x[3][3];

```
for (int i=1; i<=3; i++)
for (int y=1; y<=3; y++)
x[i-1][y-1] = pow(i, y);
for (int x=1; x<=3; x++)
for (int m=1; m<=3; m++)
cout << x[x-1][m-1] << "\t";
```

```
cout << endl;
```

1	1	1
2	4	8
3	9	27

كتابة برنامج لطباعة مصفوفة 3x3 والترتيب والتكبير

⑥ float x[3] = { 1, 2, 3, 4 }; → syntax error

⑦ float x[3] = { 1.3 } →

1.3	0	0
-----	---	---

⑧ int x[3] = { 1, 2, 3 };

```
cout << x[3]; → Logical error
3 = index out of range
```

index	0	1	2
	1	2	3

27

Array of char :-

* هو نوع خاص من ال Array .

* يمكن تعريفها إما بال char أو بال string .

① $\text{char } x[5] = \{ 'h', 'e', 'l', 'l', 'o' \};$

h	e	l	l	o
---	---	---	---	---

 *

② $\text{char } x[] = "hello";$

h	e	l	l	o	\0
---	---	---	---	---	----

* عند تعريفها بال string يبين index بعد آخر index وتحتفظ داخلها (\0) (null value)

* عدد العناصر إذا تم التخزين = عدد العناصر إذا تم تخزين قيم char متفرقة + |

* يمكن التخزين داخل 2D

* يمكن طباعة array of char مرة واحدة ← `cout << x` ويمكن عمل cin بجملة واحدة أيضاً .

① $\text{char } x[5] = \{ 'h', 'e', 'l', 'l', 'o' \};$

```
cout << x << endl;
cout << x[0] << endl;
cout << x[4] << endl;
cout << x[5] << endl;
```

hello
h
o
Logical error

② $\text{char } x[] = "Hello";$ →

H	e	l	l	o	\0
---	---	---	---	---	----

 → null value

③ $\text{char } x[2][4] = \{ "hii", "hii" \}$ →

h	i	i	\0
h	i	i	\0

String Function :-

- ① strcpy ② strcmp ③ strlen ④ cin.get

① strcpy :-

* نقل سطر s_2 الى s_1 : $strcpy(s_2, s_1)$

* نقل على نسخ قيمة ال variable الثاني ال variable الاول

② strcmp :-

* نقل سطر s_1, s_2 : $strcmp(s_1, s_2)$

* نقل على المقارنة بين قيم ال من التغييرين

* اذا كان $s_1 > s_2$ ← 1

* $s_1 < s_2$ ← -1

* $s_1 = s_2$ ← 0

③ strlen :-

* نقل سطر s_1 : $cout << strlen("-----");$

* $string s_1 = "-----"; cout << strlen(s_1);$

* بعد الحروف دون (null value)

* بعد ال white space بين الكلمات ("hi every one") ← 12

* موجود داخل $include <string>$ ← * (مطلوب)

④ cin.get (s1, 5)

string ← اسم ال string ← عدد عناصر ال string

* لاجل ادخال لقيم string

```

① string s1;
   string s2;
   string s3 = "hello";
   strcpy (s1, s3);
   strcpy (s2, "hi");
   cout << s1 << s2 << s3;

```

```

hello hi hello

```

```

② string s1 = "hello";
   string s2 = "heo";
   cout << strcmp (s1, s2);
   cout << strcmp ("hi", "hi");

```

```

-1 0

```

```

③ cout << strlen ("hi");
   cout << strlen ("hi everyone");
   string s1 = "hello";
   cout << strlen (s1);

```

```

2 11 5

```

Notes :-

* بس يكون في زيادة في عدد ال index على عدد العنصر، المخطط كان
 يعبره الباقي أرقام ، في array of char ، في العنصر اللي انزلوا بدين
 اذا كان string بجز index لا null value و الباقي بديل فراغات
 بيجيبه أرقام .

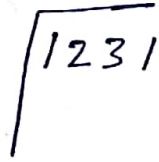
* لا يجوز وضع متغير كعدد لا index $int x=3; int E[x]=\{1, \dots, 2\};$
 ← syntax error ، إلا في حالة واحدة ان يكون المتغير x ← const
 في هذه الحالة يأخذ قيمة x كعدد لا index دون ان يعرف ← syntax

* ال null value اذا كتب داخل متغير ال array يوقف التخزين
 عنه و يغير ما بعده من متغير .


```

① int y = 0;
   const int x = 3;
   int z[x] = {1, 2, 3};
   cout << z[0] << z[1] << z[2];
   cout << z[y];

```



* ندرکز سٹوی ہون ، اجنا حکینا ال syntax اذا استخینا
 y فی ۱۲۳۱ int میں کوریم لہ index فی ۱۲۳۱ cout ندرکز ال .

```

② char b[] = "hello \0 hi";

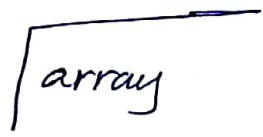
```



```

③ char b[] = "array example";
   b[5] = '\0';
   cout << b;

```



Function with Array :-

* یتیم استعمال فی ال prototype علی شکل int[]
 * عند کتابتہ فی ال call یجب اسم المتغیر فقط x, a و ہونا
 * استعاس ال array داخل ال fun یون call by reference
 * انہ ای تغیر علی قیمة ال array فی ال fun - کت علی ال array
 * فی ال main

* یعتبر call by reference → by default اذا تم استعاس ال array کامل
 * دفرة واحدة

* اذا تم استعاس index معينة من ال array by default call by value
 * لانها قیمة عادیة و یجب الاشارة ال reference فی حال عملها
 * (int &c) ← call by reference

* فی ال array 2D یمكن من اذواج ناقصة والاستعاس
 * علیها بظفار ، لولا اذا زادت ← syntax error

```

int b[2][2] = { (1,2,3), (1,2) } → X

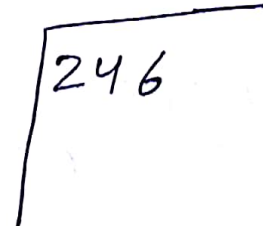
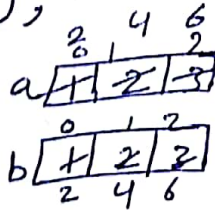
```

* اذا عرفت array ولكن لم اعطيه قيم وطبقة Logical error
 اما اذا اعطيه قيمة واحدة على الاقل يعيد الباشي ارقام

* في 2D array ← يمكن ترك خانة، لمحتوف فارغة و ملو خانة
 الاعددة لانها تحدد شكل، لمحتوفه ولكن لا يمكن ترك خانة الاعددة
 فارغة لانها يمكن ملؤها بأشكال مختلفة

```
int b [ ] [ 2 ] ; ✓
int b [ 2 ] [ ] ; X
```

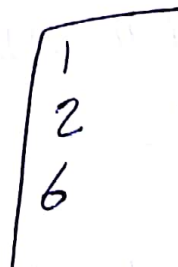
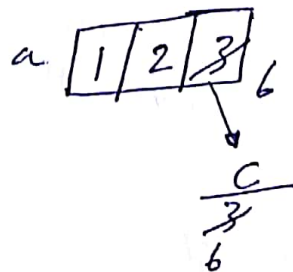
```
① void Array (int [ ] , int ) ;
void main ( )
{ int a [ 3 ] = { 1 , 2 , 3 } ;
  Array ( a , 3 )
```



```
cout << a [ 0 ] << a [ 1 ] << a [ 2 ] ; }
void Array ( int b [ ] , int array size )
{ for ( int i = 0 ; i < array size , i++ )
  b [ i ] = b [ i ] * 2 ;
}
```

ما في
طابع

```
② void Array ( int , int & ) ;
void main ( )
{ int a [ 3 ] = { 1 , 2 , 3 } ; → call by referance .
  Array ( a [ 1 ] , a [ 2 ] ) ; → call by value .
  for ( int i = 0 ; i < 3 ; i++ )
  cout << a [ i ] << endl ;
}
void Array ( int b , int & c )
{ b = 5 ;
  c = c * 2 ;
}
```



③ `int b[2][2] = { (1,2), (3,4) };`

`cout << b[0][0] << endl;`

`cout << b[1][1] << endl;`

	0	1
0	1	2
1	3	4

output
1
4

④ `int b[2][2] = { (1,2,3), (1,2) };`

↳ syntax error

اقل عادی اکثر لا

⑤ `int b[2];`

`cout << b[1];`

↳ Logical error.

⑥ `int b[][2] = { 1, 2, 3, 4 };`

1	2
3	4

انما يعرف انو
عويين فببدا
اعين فببدا

Size of Array :-

* حساب size of array ← عدد ال index * $\frac{\text{data type}}{\text{نوع}}$

↓
نوع

* لمعرفة عدد ال index من ال size

$$\text{number of element} = \frac{\text{size of Array}}{\text{size of data type}}$$

① `int x = 5;`

`int A[5];`

`int Z[2][2];`

`cout << size of (x) << endl;`

`cout << size of (A) << endl;`

`cout << size of (Z) << endl;`

`cout << size of (x[1]) << endl;`

4
20
16
4

② `int Array[] = { 1, 2, 3 };`
آلة البرنامج بحيت يتفهم عددهم

`int y = size of (int);`
`int x = size of (Array);`
`cout << x/y;`

Notes:

① لشيء لنقل قيم من array الى array آخر فلنا نستخدم
for فيه لنقل ولا يجوز `y=x;` ← `xxx`

② `cin >> array of char` فيه نقرأ لشيء
ال white space ما بعدها .

① `int x[3] = { 1, 2, 3 };`
`int y[3];`
`y = x;` → خطأ

② `char z[5];`

`cin >> z;` // hi everyone → ما تم ادخاله

`cout << z;`

hi

ولتعلم أن ما أنت ساع إليه هو ساع إليك

مع تمنياتي بالتوفيق والنجاح سامعون

وعنا عمل بشري نافع

مرور علوم

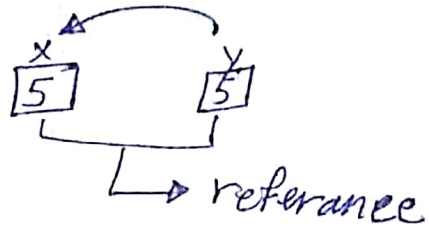
هندسة حاسوب

Pointer

~~float~~
 $\&x$ and $\&y \leftarrow \&$:: address

```

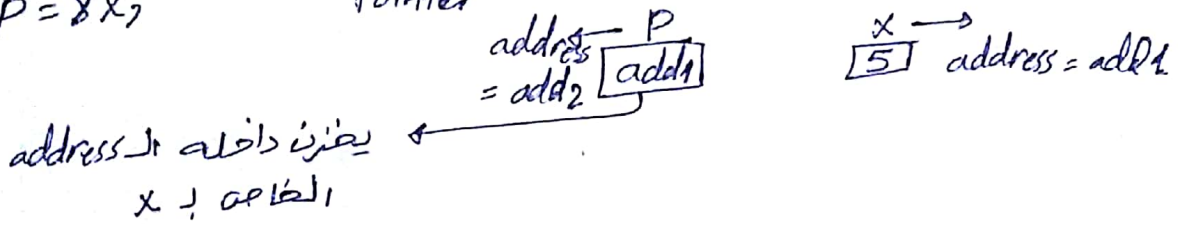
1 float x = 5;
  float y = &x;
  add us → hx05A2A8
    
```



```

2 int x = 5;
  int *p = &x;
    
```

المتا، لوجا
 Pointer



```

* int x = 5;
  int &y;
    
```

reference فيه
 لا يجوز

```

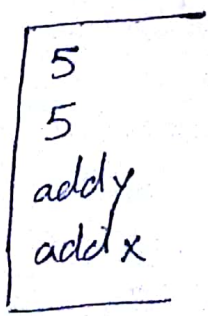
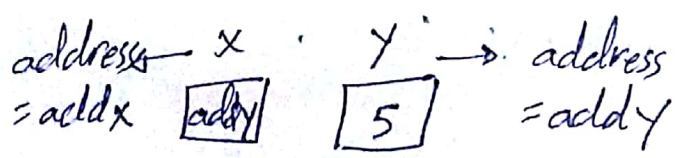
* int x = 5;
  int *p;
  p = &x;
    
```

address فيه
 يجوز فيه كتابة

```

* int y = 5;
  int *x = &y;
  cout << y << endl;
  cout << *x << endl;
  cout << x << endl;
  cout << &x << endl;
    
```

المتا لوجا ال المتا ال
 يتغير على
 المتا لوجا ال المتا ال
 يتغير على



لازم يكونوا المتتبعين من نفس النوع

* int *x = 5; X

* Pointer يعطى على address لنا لا يجوز تخزين قيمة بداخله ← syntax error

* يمكن تخزين فقط ال (0) لأنه يشير إلى ال null value لأنه لا يشير إلى شيء ← يعني خاصي

* int y = 5;
int *x = y + 5; X syntax error

* ال الاخطاء التي تشمل ال reference هي نفسها من ال address.

* int y = 10;
cout << *y;
↳ syntax error (*y ≠ y)

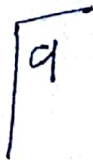
* int x = 10;
int *y = x + 1;

addx	x
add x+1	x+1
add. y	y

* لانها ليست على حسابي
في syntax error

* بعد ال add اخطاء
بـ x و يأخذ ال add التي
بعده ويعتبه في *y

* int *p;
int v = 9;
p = &v;
cout << *p << endl;



* int x = 10 ;

int *y = &x + 1 ;

cout << x << endl ;

cout << &y << endl ;

cout << *y << endl ;

cout << * &y << endl ; → Logical error

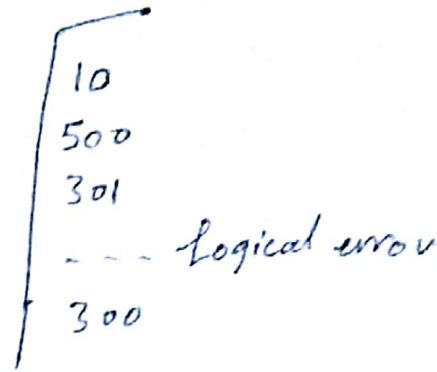
cout << &x ;

address x

300 10

301

500 y → 301



يظهر قيمة x ← cout << x ; *

يظهر الـ address الخاص بـ x ← cout << &x ; *

يظهر القيمة المتخزينه الذي يشير اليه الـ pointer ← cout << *x ; *

* char x ;

char *p = &x ;

double y = 0 ;

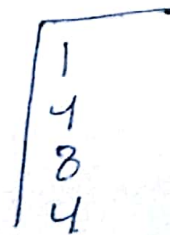
double *py = &y ;

cout << size of (*p) << endl ;

cout << size of (p) << endl ;

cout << size of (*py) << endl ;

cout << size of (py) << endl ;



* حجم الـ add = y

* int x = 5 ;

int *p = &x ;

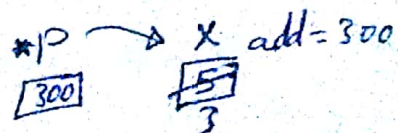
*p = 3 ;

* قيمة p هي الـ address من قيمة x

*p هي قيمة x التي تم تخزينها

قيمة *p هي قيمة x

3



const with pointer :-

* int x = 10 ;

int y = 10 ;

const int * const p = &y ;

* اذا كانت const بين * و P لا يجوز تغيير المكان الذي يوضع عليه

P = &x ; → syntax error . XXX

* اذا كانت const قبل int لا يجوز تغيير القيمة التي يوضع عليها .

* P = 5 ; → syntax error XXX

Array with pointer :-

* int x[5] = { 2, 3, 5, 4 }

int * P₁ = x ;

int * P₂ = &x[1] ;

cout << x ;

cout << * P₁ ;

cout << * P₂ ;

0	1	2	3	4
2	3	5	4	0

add → 200 201 203 210 213

≡ int * P₁ = &x[0]

اذا ما كتبت هذا اسطر
يؤخذ اول قيمة

x = address x[0]

P₁ →

x[0]
2

P₂ →

x[1]
3

output
200
2
3

* double x[5] = { 1.2, 1.3, 1.5, 1.6 } ;

int *p = x ;

p++ ;

cout << *p ;

cout << ++p ;

cout << *p ;

0	1	2	3	4
1.2	1.3	1.5	1.6	0

~~add x[0]~~
~~add x[1]~~
~~add x[2]~~
add x[3]

add x[1]
add x[?]]
1.6

Function with Pointer :-

* void fun1 (int *p)

{ cout << *p ; *p = 9 ;
cout << *p ;

}

void fun2 (int &p)

{ p++ ;
cout << p ;

}

void main ()

{ int a = 3 ;
fun2 (a) ;
fun1 (&a) ;
cout << a ;

}

4
4
9
9

