الموضوع: ____________ التاريخ: / /

Lec 4

Binary shift :-

Types shift :-

-logical shift

Transfer 0 Through The serial input Left, right

1- logical shift Left

The shift Left operation shifts each bit one place to the left

يعني لو أنا عندي char مثلا متخزن فيه 29 الهو بالبايناري
↳ 8 bits

MSb LSb

$\boxed{0}$ 0 0 1 1 1 0 1 → 29

بتتحذف

لو جيت عملتها shift left
هتبقى بالشكل دا

0 0 1 1 1 0 1 0 → 58

عملت إزاحة للـ bits شمال وضفت صفر من عاليمين

2- logical shift right

عكس الـ shift left ← بعمل إزاحة لليمين وأضيف صفر عالشمال

ولكن مثلا

MSb LSb

52 → 0 0 1 1 0 1 0 $\boxed{0}$ → بتتحذف

26 → 0 0 0 1 1 0 1 0

-The shift right operation shifts each bit one place to the right

الموضوع: ____________ التاريخ: / /

* في عملية ال shifting لازم اخلي بالي من عدد البتات 8 > 16
لأن عملية ال shift هتأثر على قيمة المتغير البعمله shift

* لو عايز انفذ عملية ال shift كـ برمجة ككود بستخدم ال Bitwise shift

* كدا انا بعمل Bitwise shift Left   29 << 1
لل 29 فهي هتتحول الأول لباينري
وبعدين يتعمل لها shift left مره واحدة

* في عملية ال shift left الرقم البعمله shift
بيضرب 2* لو بعمل shift مرة واحدة
لو بعمل shift left أكتر من مرة
29 << 2
هعمل shift left على مرتين أو اضرب الرقم
$\underset{8}{}\quad \underset{2}{}$ عدد ال shift

* $29 * 2^2 = 116$
* $29 << 2 \rightarrow 58 << 1 \rightarrow 116$

* بالنسبة لل Bitwise shift right
52 >> 1   هنا هقسم على 2 فهيبقا الناتج 26

طب لو عايز اعمل shift لعدد فردي مثلا 21
21 >> 1   المفروض الناتج 10.5 هنا بقا بهمل
النص و الناتج هيبقا 10 بس

* ليه بنقسم أو بنضرب 2* ؟ عشان الأرقام بتتحول لـ binary و ال binary
الأساس بتاعه 2 ← base 2

الموضوع: ________ التاريخ: / /

- Arithmetic Shift

→ Shift a signed number

← يعنى الرقم الى هعمله shift يعتبر اخر بت فيه هى بت الإشارة

→ Shift with sign exention → احافظ على الإشارة.

- arithmetic shift right

أخر بت بتفضل زى مهى ويعمل shift جهة اليمين مثال

```
MSB                       LSB
1  1  1  1  1  1  0  0        -4
1  1  1  1  1  1  1  0        -2
```

هنا عمل shift لآخر بت ونزلها مكانها برضه عشان يحافظ على الأشارة

* طب لو عندى بالشكل دا

```
3 ← 0000   0011
1 ← 0000   0001
```

هنا كانت 3 بقت 1 هنا عملت round لـ 1 عشان موجب

```
-3 ← 1111   1101
-2 ← 1111   1110
```

هنا كان عدد سالب فحصل round للصغير بقا بـ -2

- arithmetic shift Left

نفس البيحصل ع ال Logical shift Left بعمل shift لليسار واحد

صفر من عاليمين

الموضوع: التاريخ: / /

فالمحاضرة الفاتت خدنا ال Bitwise operator ولسه مكملين فيها

كنت بستخدمها في ال bit Manipulation :-

1- لو أنا عايز اعمل clear لبت معينه ← فمعادلة جاهزة بستخدمها بتوفر عليا كتير

Variable &= ~(1 << Bit_Position);

Variable = Variable & ~(1 << Bit_Position);

مكان البت ← يبدأ عد من صفر

Example:- 7 6 5 4 3 2 1 0

Variable = 1 0 0 $\boxed{1}$ 0 0 1 1

لعاوز أصفر البت دى

مكانها ← 4 ←

Variable = Variable & ~(1 << 4)

= Variable & ~(0001 0000)

= 1001 0011 & (1110 1111)

= 1000 0011

0000 0001
0001 0000

الموضوع: التاريخ: / /

٣- لو عايز اعمل set لبت معينه هستخدم المعادلة دى:

Variable |= (1 << Bit_Position);

Variable = Variable | (1 << BitPosition);

Example:-

```
             7 6 5 4   3 2 1 0
  variable = 1 0 0 1  [0]0 1 1
                 عاوز اخليها بواحد
variable |= (1 << 3)        0000  0001
                            0000  1000
variable = variable | (0000  1000)
         = 1001  0011 | (0000  1000)
         =      1001  1011
```

٤- لو عايز اعمل Toggle لبت معينه يعنى لو هى بصفر تبقا ا والعكس
هستخدم المعادلة دى

Variable ^= (1 << Bit_Position);

Variable = Variable ^ (1 << Bit_Position);

Example

```
             7 6 5 4   3 2 1 0
  variable   1 0 0 1   0 0 1 1
                 عاوز اعملها
                 Toggle → يعنى لو بصفر اخليها واحد والعكس
                                  0000  0001
variable = variable ^ (1 << 5)    0010  0000
         = 1001 0011 ^ (0010 0000)
         = 1011 0011
```

التاريخ: / / الموضوع:

operator	Description
sizeof()	Returns size of a variable sizeof (a), where a is integer will return ④ → depend on comiler
&	Return the address of a variable &a; Return the accual adress of the variable يرجع عنوان المتغير
*	Pointer to variable *a; use with Pointers
?:	conditional Expression

Scanf function:-
هي عبارة عن دالة بنستخدمها عشان استقبل داتا من المستخدم
وبنتبعها بالشكل دا

```
scanf("%d\n", &a);
```

$\underline{\%d}$ ← ممكن يبقا اى specifier تانى عادى

$\underline{\&a}$ ← المتغير المستقبل فيه الداتا ولازم يكون قبله علامه & عشان توصل لعنوان المتغير

* ممكن استقبل داتا لاكتر من متغير scanf("%d %d %d", &a, &b, &c);

الموضوع: ______ التاريخ: / /

conditional operator (? :)

is a ternary operator and it takes three operands

Variable = Expression1 ? Expression2 : Expression3
(condition)

* هنا فـ المعادلة دى لو Expression1 هنعتبر مكانها شرط لو الشرط دا إتحقق هينفذ الموجود فـ Expression2 لو متحققش هينفذ الى فـ Expression3 + الكلام دا فـ حالة عدم وجود varible بس هنا فيه Variable يبقى الراجع من المعادلة هيتخزن فـ الـ varible

Example:-

```
unsigned short Numberone = 1, Numbertwo = 1, Result = 0;

int main(){
   Result = Numberone == Numbertwo ? 0X11 : 0X22;
   Printf("Result = 0X %X \n", Result);
   Numberone = 3;
   Result = Numberone == Numbertwo ? 0X11 : 0X22;
   Printf("Result = 0X %X \n", Result);
   return 0;
}
```

```
Result = 0X11
Result = 0X22
```

الموضوع: ________ التاريخ: / /

operator Precedence

- decides how an expression is evaluated
- certain operators have higher Precedence Than others

معناها بشوف إيه العمليات الأولى ف التنفيذ يعني لو معادلة فيها ضرب وجمع يبقا هنفذ الضرب الأول وبعدين الجمع العملية دي كلها اسمها Precedence

operator Associativity

- is used when two operators of same Precedence apper in expression
- Associativity can be either Left to right or right to Left

معناها لو أنا عندي ف المعادلة ضرب وقسمة مثلا فأنا هنفذ الهيجي الأول من جهة الشمال أي كان هو اي العملية دي اسمها Associativity

- operators Precedence and Associativity are two characteristics of operators that determine the evaluation order of sub expression in absence of brackets.

الموضوع: ________ التاريخ: / /

Category	operator	Associativity
PostFix	() [] -> . ++ --	L to R
unary	+ - ! ~ ++ -- (type)* & sizeof	R to L
Multiplicative	* / %	L to R
Additive	+ -	L to R
shift	<< >>	L to R
Relational	< <= > >=	L to R
Equality	== !=	L to R
Bitwise AND	&	L to R
Bitwise XoR	^	L to R
Bitwis oR	|	L to R
Logical AND	&&	L to R
Logical oR	||	L to R
conditional	?:	R to L
Assignment	= += -= *= /= %= >>= <<= &= |= ^=	R to L
comma	,	L to R

الموضوع: ________ التاريخ: / /

Notes:-

- An expression doesn't get evaluted inside sizeof operator

* لو فيه Assignment جوا ال sizeof مش بتتحسب

* فيه فرق بين ← int i = 1, 2, 3; → syntax error

```
int i;
i = 1, 2, 3;      → هنا قيمة ال i هساوي 1
  ↳ (i=1), 2, 3;
```

عشان ال = أعلى من comma

int i = (1, 2, 3) ← هنا ال i هساوي 3
عشان () أعلى من = وال ()
جواها commas بيقا هيبدأ من الشمال لليمين

* لو انا كتبت كدا printf("%d", 1<<2+3<<4);
هينفذ + الأول وبعدين ال Shift وهيبدأ من الشمال لليمين

1 << 5 << 4 → 32 << 4 → 512

* x = 2+3; ، x = f1() + f2();
هنا عليا الحج تتعمد على الكمبيولر compiler سواء هيجمع هين مع هين الأول يعني 2^{nd} 3 أو 2^{nd} 3

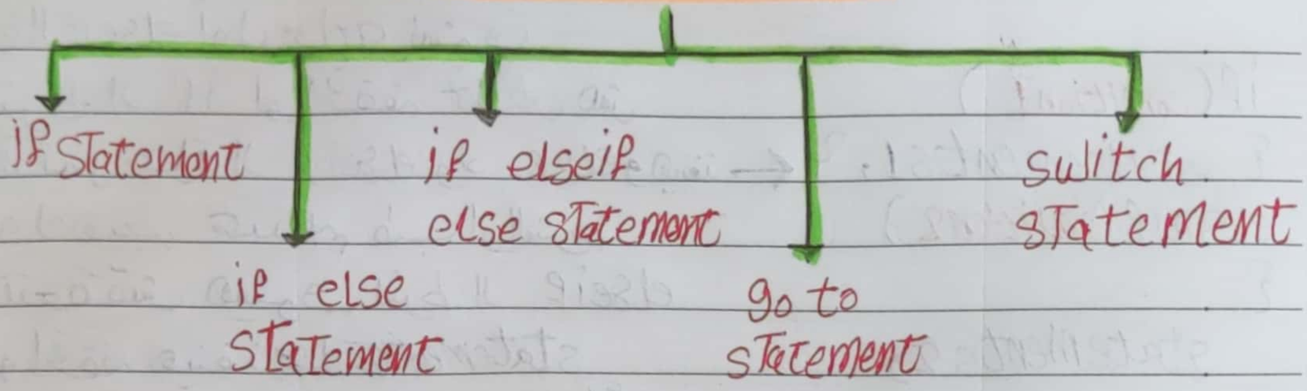
* ال يساوي أعلى من ال comma عشان كدا ال y=12

$\boxed{13 \quad 12}$

```
int x = 10;
int y = (x++, x++, x++)
printf("%d %d", x, y);
```

الموضوع: ________ التاريخ: / /

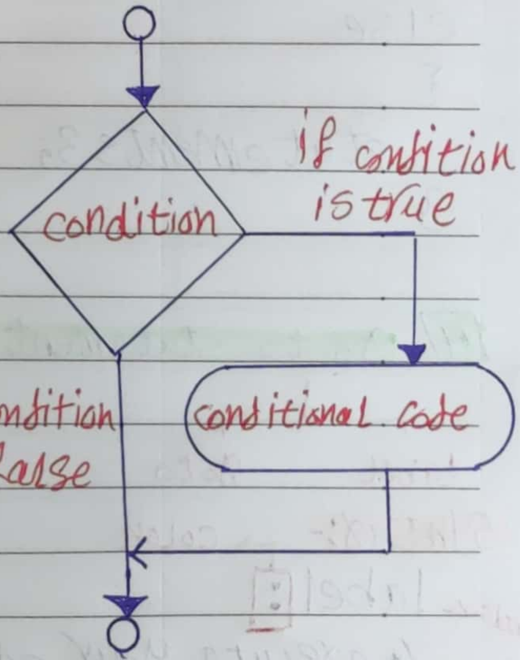
Decision Making



1 if statement :-

condition ⟶ ممكن يبقا أكتر من شرط

```
if( الشرط )
{
  statement 1;
  statement 2;
}
```

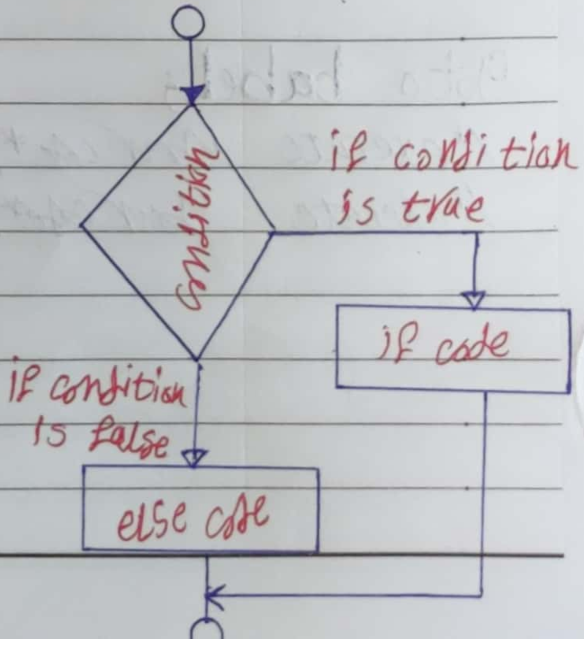


* جملة if بتتنفذ إزاى ؟!
لو الشرط إتحقق وبقى True هينفذ
الجوا { } أى كله هواى
لو الشرط متحققش وبقى false هيروح
ع ال statement ال بعد ال if

* لو مفيش { } والشرط إتحقق هيتنفذ أول جملة بعد if بس

2 if else statement :-

```
if( condition )
{
  statements;
}else
{ statements
}
```



* لو ال condition ميتحققش هينفذ الجوا else لو إتحقق هينفذ ال if

3 else if statement :-

```
if(condition1 )
{ statements1; }
elseif (condition2 )
{
  statements 2;
}
else
{
  statements 3;
}
```

* في ال elseif أول حاجه بيشوف
شرط ال if لو اتحقق خلاص مش
هيكمل على ال elseif ولا else وهينفذ ←
ولا هيجي عندهم طب لو الشرط
متحققش هيشوف شرط ال elseif
لو اتحقق هينفذ جملة statements2
لو متحققش هينفذ ال else تلقائي.

* يفضل تكتب ال else حتى لو هتسيبها فاضيه
* في حالة وجود أكثر من شرط يفضل اخلي
كل شرط لوحده بين ()

4 go to statement :-

label goto

Syntax:- → colon

```
عنوان ← label :
  /* execute your code */
  /* execute your code */

goto label ;
  /*execute your code*/
  /*execute your code*/
```

ال label دا ممكن يبقا اي
اسم بس مش من ال keywords

* مش محبب استخدامها لانه
البروسيسور بيقفز من مكانه
لمكان ومن مكانه لمكان
تاني وهكذا

* تنفيذ البرنامج أول ميلاقي
goto هيروح للمكان الجمبها
وينفذ ال label عند دا

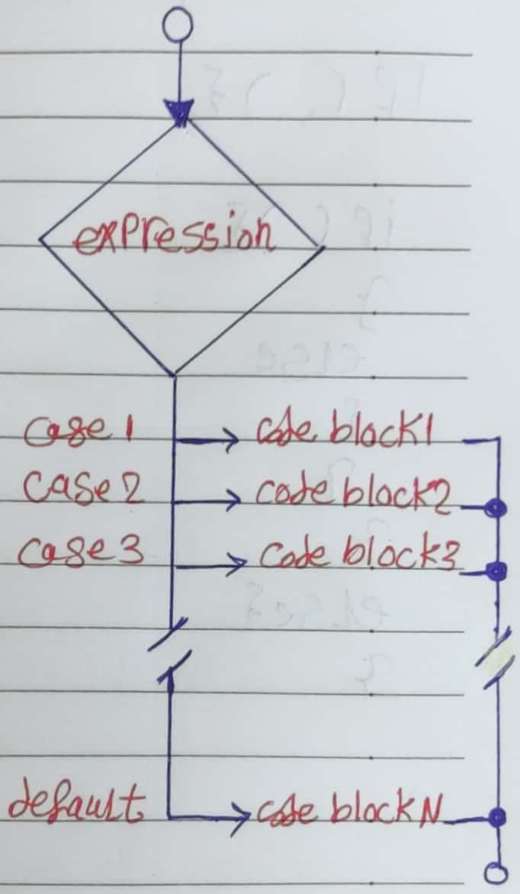
الموضوع: ________ التاريخ: / /

5 Switch statement :-

بستخدم لو انا عندي variable ليه أكتر من حالة وأنا عارف الحالات دي

Syntax:-

```
Switch(variable)
{
    case 1 :-
     /*execute your code*/
     break;
     Case 2:
     /*execute your code*/
     break;
     Case n:
     /*execute your code*/
     break;
   default:
     /*execute your code*/
     break;
 }
```



* الـ switch بتشوف الـ variable أي كان نوعه رقم أو حرف شبه أنهي case عندك أو بتساويه ويتروح تنفذ الـ statement الـ ئد الـ case المشابها لو ملقاش فيه case مشابها للـ variable بتنفذ الـ statements الـ ع الـ default + الـ default ممكن تبقا في أي مكانه داخل الـ switch بس يفضل يبقا معاها break في أي مكانه جوا switch

* الـ break بتخرجني من الـ switch ولو مش موجودة هتنفذ الـ cases الـ مفهاش break

التاريخ: / /

الموضوع:

Nested if else

```
if (  ){

  if ( ){
  }
   else
   {
   }
}
else{
}
```

Nested switch

```
switch(){
 case :
  switch(){
   case :
    break;
    default:
    break;
   }
  case :
  break;
  default:
   break;
}
```