c programming
Written By The-oNe
Part 3

بسم الله الرحمن الرحيم

- الإتفاقية:

لقد كتبت هذا الملف لغرض تثقيف الشباب العربي في مجال علوم الحاسب الآلي. وهذا الملف مجاني للجميع ولا أريد من ورائه إلا شيء بسيط جداً وهو دعوة صالحة في ظهر الغيب لي ولجميع أخواننا المسلمين في أنحاء العالم.كما أرجوا أن لا يتم التعديل في هذا الملف وإنسابه إلى غيري لأني قد تعبت فيه كثيراً. كما أنني أؤؤكد أنني أرحب وبكل سعة صدر بالنقد البناء الذي يستفيد منه الجميع. كما أنه إذا وجدت عزيزي القارئ أي أخطاء في هذا الملف يرجى أخباري بها وسيتم وضع أسمك في هذا الملف مع التعديل كما أني أرحب أن يشترك معي أي شخص لإضافة المزيد من الدروس لهذا الملف وسيتم وضع أسمه أيضاً وذلك حتى يكون هناك مرجع عربي للغة السي.

- التعليمات :

١. التعليمتان (++) و (--) التي تلي المتغير:

وهذه التعليمتين تأتي بعد المتغير. والصورة العامة لها هي :

```
Operand ++;
Operand --;
```

وهي تعني أن القيمة لهذا المتغير سوف تزيد بمقدار واحد إذا كانت التعليمة (++) أو أنها ستنقص بمقدار واحد إذا كانت التعليمة هي (--).
لاحظ المثال التالي:

```
Int a=5;
a++;
Printf("%d",a);
```

لاحظ أننا في المثال السابق لم نكتب برنامجاً كاملاً وذلك لغرض عدم الإطالة في الدرس. لاحظ من المثال السابق سوف يكون ناتج البرنامج هو العدد (6) وذلك لأننا في البداية أسندنا قيمة المتغير إلى العدد (5) ولكننا إستخدمنا التعليمة (++) لكي نزيد قيمة المتغير (a) بمقدار واحد. أي أن التعبير (a++) يكافئ التعبير (a=a+1).وينطبق نفس الكلام السابق على التعليمة (--) .

٢. التعليمات (++) و (--) التي تسبق المتغير:

ونلاحظ أن التعليمتان سبق وأن تعرفنا عليها قبل قليل ولكن التعليمتين السابقتين كانت تستخدم بعد المتغير أما هنا فستنستخدمها قبل العامل على الصورة التالية:

```
++operand;
--operand;
```

لاحظ المثال التالي:

```
Int b=4;
Printf("%d",++b);
```

لاحظ في هذا المثال السابق أنه من المفترض أن يتم طباعة العدد (4) وذلك لأن قيمة المتغير هي (4) ولكن الناتج سوف يكون هنا (5) وذلك لأنه سوف يتم زيادة قيمة المتغير قبل أن يتم طباعته أي أن محتوى المتغير (b) في الذاكرة سوف يتغير من القيمة (4) إلى القيمة (5) وسيتم تخزينها في هذا المحتوى ثم بعد ذلك طباعتها. ونفس الكلام أيضاً ينطبق على التعليمة (--).

٣. التعليمة (!):

وهذه التعليمة تدل على القيمة المخزنة في الذاكرة لهذا المتغير سوف تعطى نفيها أي إذا فيمة المتغير هي (1) أي بمعنى (true) بعد إستخدام هذا العامل سوف تكون قيمة المتغير هي (0) أي بمعنى (false) . والصورة العامة لهذه التعليمة هي (!operand). لاحظ المثال التالي:

```
Int a=1;
Printf(“%d”,!a);
```

في المثال السابق سوف يتم طباعة القيمة (0) للمتغير (a) وذلك لأننا إستخدمنا التعليمة (!).

٤. التعليمات الحسابية المعروفة (+ - * / %):

ونظراً لأن هذه التعابير تعتبر تعابير معروفة لدى الجميع فإني سأكتفي فقط بمثال :

```
#include<stdio.h>
main ()
{
int a=10, b=5;
printf("%d\n",a+b); // ١٥ النتيجة ستكون هنا
printf("%d\n",a-b); // ٥ النتيجة ستكون هنا
printf("%d\n",a*b);  // ٥٠ النتيجة ستكون هنا
printf("%d\n",a/b);  //٢ النتيجة ستكون هنا
printf("%d\n",a%b);  //٠ النتيجة ستكون هنا
return 0;
}
```

في المثال السابق نلاحظ أنه تقريباً جميع العمليات معروفة لدى الجميع ماعدا العملية (%) وهي تعني باقي القسمة أي أنه إذا قسمنا العدد (10) على العدد (5) سوف يكون الناتج (2) والباقي هو (0) وذلك لأن (10/5=2). والجدول التالي يبين متخصر البرنامج السابق:

اسم العملية	رمز العملية	مثال على العملية
الجمع	+	A+b=10+5=15
الطرح	-	a-b=10-5=5
الضرب	*	A*b=10*5=50
القسمة	/	A/b=10/5=2
باقي القسمة	%	A%b=10%5=0

٥. التعليمات التي تعمل على مستوى البت (bitwise):

قبل أن نبدأ في هذه التعليمات لابد من معرفة أن البت هو أصغر وحدة تخزين في الذاكرة والبايت هو عبارة ثمان بتات أي أن (1 byte = 8 bit) حيث أن أكبر قيمة في البايت هي القيمة (2^8-1).

- التعليمة (~) :

وهذه التعليمة تقوم بعملياتها على مستوى البت (bit) وهي تقوم بإعطاء القيمة (0) إذا كانت قيمة هذا البت تساوي (1) وتعطي قيمة هذا البت (1) إذا كانت قيمته تساوي (0). مثلاً لو أردنا تمثيل العدد (5) في الذاكرة طبعاً ستكون قيمة العدد (5) هي قيم عبارة عن (1) أو (0). لاحظ تمثيل العدد :

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	0	1	0	1

لاحظ التمثيل التالي بما أن قيمة العدد هو (5) إذن (2^0=1) و (2^2=4) ولهذا وضعنا في الخانتين (2^0) و (2^2) القيمة (1) حتى يتسنى لنا تمثيل العدد (5). لاحظ أنه إذا تم تطبيق التعليمة (~) هنا ستتحول القيم الصفرية (0) إلى (1) والعكس صحيح.

- التعليمة (&):

وهذه التعليمة أيضاً تقوم بعملياتها على مستوى البت وهي تعني (and) والصورة العامة لها :

Expression1 &expression2

وهذه التعليمة تعطي القيمة (1) إذا كان البتين في القيميتن العدديتين يساوي واحد وإلا فإنها تعطي القيمة واحد ونستطيع أن نوجد لهذه العملية جدول بالقيم التي يتوقع أن تظهر لنا ويمكن تسمية هذا الجدول بجدول الصواب :

Expession1	Expression2	Result
0	0	0
0	1	0
1	0	0
1	1	1

أي لاحظ أنه لا يتم إعطاء القيمة (1) إلا إذا كان البتين في التعبيرين هو (1) لا حظ المثال التالي:

```
#include<stdio.h>
main ()
{
int a=5,b=4;
printf("%d",a&b);
return 0;
}
```

في المثال السابق سيصبح الناتج هو (4) وذلك للسبب التالي:

Expression1(a)	0	0	0	0	0	1	0	1
Expression2(b)	0	0	0	0	0	1	0	0
Result	0	0	0	0	0	1	0	0

لاحظ أن البت الثالث فقط هو الذي يكون فيه التعبيرين يساوي (1) لذلك فإن القيمة المعطاة هنا حسب جدول الصواب هي (1) وبما أن البت الثالث هو عبارة عن (2^2=4) إذن الناتج سيظهر أربعة كما ذكر.

- التعليمة (|):

وهذه التعليمة هي عبارة عن (OR)وهي تعطي القيمة (0) إذا كان قيمة البت نفسه في المتغيرين هو (0) وجدول الصواب لها هو كما يلي:

Expression1	Expression2	Result
0	0	0
0	1	1
1	0	1
1	1	1

وينطبق على هذه التعليمة جميع ما سبق ذكره في التعليمة (&).

- التعليمة (^):

وهي تعني (XOR) وهذه التعليمة تعطي القيمة (0) إذا تشابه البت في نفس التعبيرين . وجدول الصواب لها هو كما يلي:

Expession1	Expression2	Result
0	0	0
0	1	1
1	0	1
1	1	0

وأيضاً ينطبق عليها ما سبق ذكره في التعليميتن (&) و (|).

- تعليمة الإزاحة لليمين (>>):

وهذه التعليمة تعمل أيضاً على مستوى البت والصورة العامة لها هي :

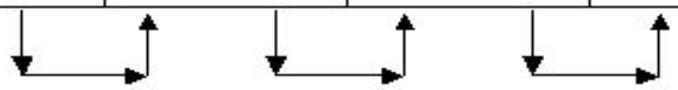
```
Expression >> shift_value
```

حيث أن (shift_value) هي عبارة عن عدد البتات التي سيزاح إليها العدد.

```
Int a=5;
Printf("%d",a>>1);
```

في هذا المثال نلاحظ أن قيمة المتغير هي (5) وبما أن هذه القيمة تعمل عادةً على مستوى البت فلابد أنها تعتمد على التمثيل الثنائي للعدد (5) وحيث أن العدد (5) بالتمثيل الثنائي هو (0101) إذن المثال السابق سوف يعمل على هذا العدد (لاحظ أننا لم نذكر الأصفار التي تسبق العدد من ايسار وذلك لأن لا قيمة لها حتى وإن أضيفت) الآن لكي تفهم معي هذا المثال لاحظ الشكل التالي:

2^3	2^2	2^1	2^0
0	1	0	1



لاحظ أن البت (2^1) سوف يحل محل البت رقم (2^0) والبت (2^2) سوف يحل محل البت رقم (2^1) وهكذا إلى أن يتم إزاحة آخر بت في التمثيل وهو البت (2^3)و سيصبح شكل العدد بعد إجراء عملية الإزاحة عليها هي :

2^4	2^3	2^1	2^0
0	0	1	0

أي أن قيمة المتغير سوف تصبح (2).إذن سوف يكون ناتج البرنامج السابق هو(2).

- تعليمة الإزاحة لليسار (<<):

وهذه التعليمة تعمل أيضاً على مستوى البت والصورة العامة لها هي :

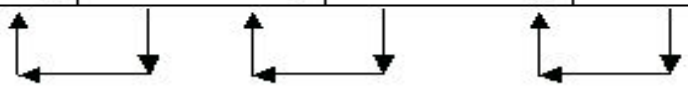
```
Expression <<shift_value
```

حيث أن (shift_value) هي عبارة عن عدد البتات التي سيزاح إليها العدد.لاحظ المثال التالي:

```
int a=5;
printf("%d",a<<1);
```

في هذا المثال سوف يصبح الناتج هو (10) وذلك حسب الشكل التالي:

2^3	2^2	2^1	2^0
0	1	0	1



لاحظ هنا أنه سوف يتم إزاحة البت (2^2) إلى البت رقم (2^3) وكل بت للذي يليه وسيصبح الشكل بعد تنفيذ عملية الإزاحة هي :

2^4	2^3	2^1	2^0
1	0	1	0

أي أن ناتج الإزاحة سوف يكون (1010) وهو يساوي العدد (10) بالنظام العشري أي أن ناتج هذا المثال هو العدد (10).

- التعليمات المنطقية:

التعابير المنطقية هي التعابير الذي يكون ناتجها أما صح أو خطأ أي لا تحتمل أكثر من قرار والجدول التالي يبين التعابير المنطية في لغة السي:

رمز العملية	اسم العملية	الصورة العامة لها
&&	(and) أي بمعنى (و) في اللغة العربية	Expression1 && expression2

||	(OR) أي بمعنى (أو) في اللغة العربية.	Expression1 || expression2
!	(not) أي بمعنى النفي .	!expression

وهذه التليمات لها نفس جداول الصواب للتعليمات (|) و (&) أما جدول الصواب للتعليمة (!) هو يحتمل على قرارين فقط :

Expression	Result
1	0
0	1

- تعليمات المقارنة:

وهذه التعابير تستخدم للمقارنة بين قيمتين لمتغييرين ومن هذه التعابير ما تلاحظه في الجدول التالي :

رمز التعبير	معناه
==	أي أن القيم في المتغير الأول تساوي نفس القيمة في المتغير الثالث ولاحظ أن هذا التعبير للمقارنة وليس للإسناد.مثل (a==b) أي أن قيمة المتغير (a) هي نفس القيمة في المتغير (b).
!=	أي أن القيمة في المتغير الأول لا تساوي القيمة في المتغير الثاني.مثل (a!=b) أي أن القيمة في المتغير (a) لا تساوي القيمة في المتغير (b).
>	علامة أكبر من ويستفاد منها في أنها تقارن بين قيمتين لمتغيرين مختلفين.مثل (a>b) وهي تعني أن قيمة المتغير (a) أكبر من قيمة المتغير (b).
<	علامة أصغر من ويستفاد منها في أنها تقارن بين قيمتين لمتغيرين مختلفين.(a<b) وهي تعني أن قيمة المتغير (a) أقل من قيمة المتغير (b) .
>=	علامة أكبر من أو يساوي وهي تستخدم في المقارنة أيضاً بين قيمتين لمتغيرين مختلفين مثل .(a>=b) وتعني أن قيمة المتغير (a) أكبر من أو يساوي للقيمة الموجودة

	في المتغير (b).
<=	علامة أصغر من أو يساوي وهي تستخدم أيضاص للمقارنة بين قيمتين لمتغيرين مختلفين مثل. (a<=b) أي أن قيمة المتغير (a) أقل من أو تساوي القيمة في المتغير (b).

وغالباً ما تستخدم هذه التعابير في التعابير الشرطية والذي سوف يتم شرحها لاحقاً لذلك سوف نؤجل أمثلتها لدرس التعابير الشرطية.

- عمليات الإسناد:

ونقصد بعمليات الإسناد أي إسناد قيمة ما إلى متغير. ومن الممكن أن تأخذ هذه التعليمات العديد من الأشكال وهي كالتالي:

التعبير العام للتعليمة	التعبير المكافئ لها	شرح عمل التعليمة
Expression1=expression2	نفس التعبير السابق	أي أن قيمة (expression2) سوف تخزن في (expression2).
Expression1+= Expression2	Expression1= Expression1+ Expression2	جمع التعبيرين (Expression1) و(Expression2) ووضع الناتج في (Expression١)

لاحظ التعليمات التالية ينطبق عليها الشرح للعتبير (Expression1+= Expression2).

- Expression1-= Expression2
- Expression1*= Expression2
- Expression1/= Expression2
- Expression1%= Expression2
- Expression1<<= Expression2
- Expression1>>= Expression2
- Expression1&= Expression2
- Expression1^= Expression2
- Expression1 |= Expression2

- الأولويات في التعليمات السابقة:

ونعني بالأولوية أي التعليمات التي سوف يتم تنفيذها قبل الأخرى لاحظ الجدول التالي:

التعليمات التي بين الأقواس	()
عملية النفي	!
تعليمات الزيادة والإنقاص	++ --
تعليمات الضرب والقسمة وباقي القسمة	* / %
تعليمات الجمع والطرح	+ -
تعليمات الإزاحة	<< >>
تعليمات المقارنة	< > <= >=
تعليمات المنطق	== !=
تعليمة (AND) على مستوى البت	&
تعليمة (XOR) على مستوى البت	^
تعليمة (OR) على مستوى البت	|
تعليمة (AND) المنطقية	&&
تعليمة (OR) المنطقية	||
تعليمات الإسناد	=

- خليط المتغيرات في التعابير :

في المثال السابق عندما قمنا بالعمليات الحسابية كانت جميع المتغيرات من نوع واحد فقط وهو متغير عدد صحيح (integers). ولكن إن كانت المتغيرات في تعبير واحد أكثر من نوع مثل جمع متغير عددي صحيح مع متغير حرفي وهكذا.والجدول التالي يبين كيفية تعامل لغة السي مع خليط المتغيرات في التعبير الواحد،لاحظ أن الذي ينطبق على عملية الجمع ينطبق على جميع العمليات الباقية.

الحالة	التغير الذي سيحصل	مثال
عندما يكون في التعبير الواحد متغير عددي صحيح (integer) ومتغير عددي حيقي بفاصلة عشرية (float).	إذا وجد في تعبير واحد هذين النوعين من المتغيرات فإن المفسر سوف يحول المتغير العددي الصحيح إلى متغير عددي حقيقي.	C= 5+3.14=8.14
عندما يكون في التعبير الواحد متغير عددي صحيح مع متغير (integer) مع متغير حرفي (	إذا وجد في تعبير واحد هذين النوعين من المتغيرات فإن المفسر سوف يحول المتغير الحرفي إلى	C=A+3=68 وأصبحت النتيجة بهذه الصورة لأن المفسر عندما يرى هذا النوع من الخليط فسوف

.(character	قيمة عددية حتى يستطيع أن يقوم بالعملية الحسابية. ويتم ذلك من خلال جدول الترميز أسكي .لا حظ المثال.	يستعين بجدول الترميز أسكي وذلك لكي يعرف القيمة العددية لهذا الحرف وبالنظر لجدول الترميز أسكي فإن قيمة الحرف(A) هي (65) إذن (65+3=68).
عندما يكون في التعبير الواحد متغير عددي حقيقي (float) مع متغير حرفي (.(character	إذا وجد في تعبير واحد هذين النوعين من المتغيرات فإن المفسر سوف يحول المتغير الحرفي إلى قيمة عددية حتى يستطيع أن يقوم بالعملية الحسابية. ويتم ذلك من خلال جدول الترميز أسكي .لا حظ المثال.	C=A+3.14=68.14 لاحظ أنه قد تم تحويل القيمة العددية للمتغير الحرفي إلى قيمة عددية حتى يتسنى للمفسر القيام بالعملية الحسابية وكما قلنا سابقاً بما أن (A) تساوي العدد (65) في جدول الترميز أسكي إذن ((65+3.14=68.14

وتقبلوا خالص تحيات أخوكم المحب (The-oNe)
الرجاء إرسال مقترحاتكم وآرائكم على العناوين التالية:
The-one@pharaonics.net
OR
The_o0ne@hotmail.com
OR
The_o0one@yahoo.com