العنوان:	تعليم البرمجة بلغة C
المصدر:	المجلة العربية العلمية للفتيان
الناشر:	المنظمة العربية للتربية والثقافة والعلوم
المؤلف الرئيسي:	حسين، نبيل
المجلد/العدد:	مج 3, ع 6
محكمة:	نعم
التاريخ الميلادي:	1999
الشهر:	ديسمبر
الصفحات:	58 - 66
رقم MD:	100448
نوع المحتوى:	بحوث ومقالات
قواعد المعلومات:	EduSearch
مواضيع:	لغات البرمجة، الحاسبات الإلكترونية، تعليم البرمجة، لغة السي C، البرمجيات
رابط:	http://search.mandumah.com/Record/100448

تعلم البرمجة بلغة C

نبيل حسين :
رئيس شركة عبور النور للبرمجيات الجاهزة - عمان - الأردن.

قد يبدو لك هذا المقال متخلفاً، فلغة Visual C++ تتربع على عرش لغات البرمجة، وأنا أكتب مقالاً تعليمياً عن لغة C! ولكن كيف تستطيع تعلم لغة VC++ من دون تعلم لغتي C و C++؟! والحقيقة أن هذه السلسلة التعليمية موجهة إلى الأشخاص الذين يريدون أن يتعلموا لغتي C و C++ بشكل سريع ومن دون الغوص في تعقيدات هاتين اللغتين ؛ لكي يستطيعوا بعد ذلك تعلم لغة VC++.

لمحة تاريخية !

لقد تم تطوير لغة C في عام 1972 في مختبرات «بل» على يد دينيس ريتشي. وتعدّ لغة C من لغات البرمجة العليا (High level language)، ومن أهم ميزاتها هو أنها قد صممت لكتابة أنظمة التشغيل وبرامج الترجمة، وغيرها، وكانت هذه البرامج تكتب من قبل باستخدام لغة الآلة (Machine language). وقد أثبتت لغة C أن البرامج المكتوبة بها تقارب البرامج المكتوبة بلغة الآلة من حيث الكفاءة.

وكان نظام التشغيل Unix دليلاً واضحاً على قدرة لغة C ؛ فقد كانت الإصدارة الثانية، التي تمت كتابتها في سنة 1972،

أغلبها مبرمجة باستخدام لغة C. ويمكن كتابة برنامج C باستخدام محررات النصوص الخاصة بالمترجمات (Compilers) ، أو أي من محررات النصوص العادية، ومن ثم ترجمة هذا البرنامج باستخدام أي من مترجمات لغة C (Compilers). وبعد كتابة البرنامج، يجب فحصه والتأكد من خلوه من الأخطاء كي يتمكن المترجم من تنفيذه . ويعرض مترجم لغة C، بالإضافة إلى الأخطاء (إن وجدت في البرنامج)، التحذيرات (Warnings). والتحذير يعني احتمالية وجود خطأ في البرنامج. فإن احتوى البرنامج على «أخطاء» فإن المترجم لن يقوم بتنفيذ البرنامج حتى يقوم المبرمج بتصحيحها، ولكن إن احتوى البرنامج على «تحذيرات»، فإن المترجم سيتمكن من تنفيذ البرنامج وسيشير إلى هذه «التحذيرات» فقط.

وقد استمرت لغة C بالتطور حتى صدرت لغة C++، ومن الشركات التي تنتج مترجمات للغة C و C++:- Inprise, Microsoft, Symantec وغيرها ...

الهيكل الرئيس للبرنامج:

يبدأ القسم الرئيس للبرنامج بالدالة "main ( )"، وتحصر الأوامر داخل قوسي المجموعة: { } ، كما في المثال الآتي:

```
main ()
{
// This is a comment.
/* This is a comment too. */
}
```

ومن الجدير بالذكر، أنه يجب التقيُّد عند كتابة البرنامج بالحروف الصغيرة والكبيرة، فإن كتبت مثلاً كلمة main بأحرف كبيرة فإن المترجم سيظهر رسالة تفيد بوجود خطأ في هذه العبارة. كما أن أوامر لغة C تنتهي بالفارزة المنقوطة (;)، مستثنين من ذلك الدالة (main)، وكذلك بقوسي البداية والنهاية ({})، وبعض الإجراءات الأخرى كما سنرى لاحقاً.

كذلك يمكن كتابة الملاحظات والتعليقات في البرنامج ولكن يجب وضع الإشارة (//) قبل الملاحظة، أو حصر الملاحظة بين العلامتين (/* */) التي يسمونها Comment أو Remark. (انظر إلى المثال أعلاه).

المتغيرات والثوابت:

أ - الأنواع (Types)

المتغير: هو عبارة عن موقع في الذاكرة، تخزن فيه قيمة نستطيع تغييرها، ويلغى هذا الموقع فور خروجك من البرنامج.

وعند تعريف متغير لغرض وضع قيمة فيه يجب أن نختار النوع المناسب للقيمة التي نريد أن نضعها في هذا المتغير، وتحتوي لغة C على أنواع عدة، منها:

Char ـ لوضع عدد صحيح من ـ 128 إلى +127

Unsigned Char ـ لوضع عدد صحيح من 0 إلى +255

int ـ لوضع عدد صحيح من ـ 32768 إلى 32767

Unsigned int ـ لوضع عدد صحيح من 0 إلى 65535

Long ـ لوضع عدد صحيح من ـ 2.147.483.648 إلى +2147483647

Unsigned Long ـ لوضع عدد صحيح من 0 إلى 4.294.967.295

Float ـ لوضع عدد حقيقي من (10^{-38}) * 3.4 إلى (10^{38}) * 3.4

Double ـ لوضع عدد حقيقي من (10^{-308}) * 1.7 إلى (10^{308}) * 1.7

Long double ـ لوضع عدد حقيقي من (10^{-4932}) * 3.4 إلى (10^{4932}) * 1.1

bool :- لوضع إحدى القيمتين المنطقيتين (true, false)، علماً بأن هذا النوع موجود فقط في المترجمات الحديثة للغة C. وهو مختصر لكلمة (Boolean).

ويمكنك استخدام كلمة "long" مع النوعين "int" و "double"، كي تتضاعف مساحتهما فتصبح في حالة الـ (int) أربعة بايتات، وفي حالة الـ (double) ستة عشر بايتاً.

وقد تتساءل، عزيزي القارىء، ما التعريف الخاص بالقيم الرمزية «الحروف»؟ إن التعريف الخاص بالقيم الرمزية سوف ندرسه في موضوع «المؤشرات»، في الجزء الثاني من هذا المقال، إن شاء الله.

ملاحظة مهمة: عندما تقوم بتعريف متغير دون أن تضع فيه أية قيمة ابتدائية، فسوف توضع فيه قيمة عشوائية.

ب - تعريف المتغيرات:

هناك نوعان من المتغيرات:

المتغيرات العامة (Global variables): وهي المتغيرات التي يمكن لأية دالة من دوال البرنامج أن تتعامل معها (أي تقرأ أو تُغيِّر قيمتها)، ولاينصح باستخدام هذه المتغيرات.

المتغيرات المحلية (Local variables): وهي المتغيرات التي ستكون معرفة فقط للجزء الذي تم تعريفها فيه. فمثلاً، إذا قمنا بتعريف المتغير "I" في الجزء الرئيس من البرنامج فلا يمكن استخدامه في أية

دالة أخرى، وإنما فقط في الجزء الرئيس من البرنامج. ولتعريف أي متغير، نكتب:

اسم المتغير نوعه ;

مثال: int i;

المثال التالي يوضح كيفية تعريف المتغيرين "I,j" كمتغيرات عامة، والمتغير "var" كمتغير محلي:

```
float I,j;
main ( )
{
char var;
}
```

ج - إعطاء المتغير قيمة:

لإعطاء أي متغير قيمة، نقوم بكتابة اسم المتغير ثم نكتب علامة المساواة (=) ثم نكتب القيمة التي نريد أن نضعها في ذلك المتغير، وأخيراً نضع الفاصلة المنقوطة. مثال:

```
main ( )
{
int I;
char ch; float j;
double d;
i = 5;
ch = 8; d = 11.2;
}
```

في المثال السابق خطأ، هل اكتشفته؟ حسناً، قلنا إن الحروف الصغيرة والكبيرة تؤثر، ونحن قد عرّفنا المتغير "I" ثم قمنا بإعطاء قيمة لمتغير غير موجود أصلاً هو " i ". كما أنه يجب كتابة الأعشار عندما نقوم بتعريف المتغير على أنه "float" أو "double". أرجو أن تلاحظ بأنك تستطيع كتابة أكثر من أمر واحد في السطر الواحد.

د - تعريف الثوابت:

يمكن تعريف الثابت (عبارة عن متغير تكون قيمته ثابتة أي لاتتغير) بكتابة التعليمة "#define" ثم اسم الثابت، ثم القيمة المراد وضعها فيه.

المثال التالي يقوم بحساب مساحة دائرة نصف قطرها ٥ سم:

```
# define pi 3.14
main ( )
{
float area = 5 * pi;
}
```

ملاحظة: يمكن إعطاء المتغير قيمة وتعريفه في خطوة واحدة، وذلك بكتابة علامة المساواة بعد اسم المتغير ثم كتابة القيمة التي نريد أن نضعها فيه؛ والقيمة إما أن تكون مباشرة مثل:

```
int I = 5;
```

أو أن تكون عن طريق معادلة رياضية كما في المثال السابق.

كذلك ويمكن استخدام الأمر "define" في تعريف مرادف لتعليمة معينة، كما في المثال التالي الذي يقوم بتعريف الكلمة (begin) لتكون مرادفة لقوس البداية، والكلمة (end) لتكون مرادفة لقوس النهاية :

```
#define begin {
#define end }
main ( )
begin
int j;
j = 5;
end
```

علامات الزيادة والنقصان المختصرة:

تعتمد لغة C على الاختصارات كثيراً، حيث تستطيع تنفيذ عمليات عدة في خطوة واحدة . ولاتقتصر الاختصارات في لغة C على هذا النوع فحسب، وإنما تمتد لتشمل حالات أخرى، منها طرق زيادة وإنقاص قيم المتغيرات.

في الحالة الاعتيادية ، نستطيع زيادة قيمة المتغير "j"، رقماً واحداً ، بكتابة المعادلة: (j = j + 1;) ، ولكننا نستطيع اختصار تلك المعادلة بكتابة: (j++;) ؛ حيث سيقوم المترجم بزيادة قيمة المتغير "j" رقماً واحداً. ونستطيع كتابة ذلك بصيغة أخرى، وهي: (++j;). ونستطيع استبدال إشارة الجمع بإشارة الطرح ليقوم المترجم بإنقاص قيمة المتغير "j" رقماً واحداً.

مثال :

```
main ( )
{
int j = 10 , x = 46 ;
char k = 8 , c = 5 ;
j++ ;
c--;
++k ;
}
```

كذلك نستطيع زيادة قيمة متغير معين، أو إنقاصه، أو ضربه، أو تقسيمه على عدد آخر، بكتابة اسم المتغير، ثم كتابة رمز العملية المطلوبة، ثم وضع علامة المساواة، ثم كتابة العدد الذي نريد أن نضيفه على المتغير، أو الذي نريد إنقاصه منه، أو الذي نريد ضربه به، أو الذي نريد تقسيمه عليه، مثال :

```
j + = 5;
c- = 3 ;
k * = 6 ;
x % = 7 ;
```

ملاحظة : تستخدم العلامة (%)، لإيجاد باقي القسمة؛ ولهذا سيخزن في المتغير "x" باقي قسمة قيمة المتغير "x" على الرقم 7.

الدوال:

أ - ماهية الدالة؟

الدالة (Function): هي ببساطة عبارة عن برنامج فرعي يقوم بعدد من العمليات ، ثم يقوم بإرجاع، أو عدم إرجاع، قيمة إلى الجزء الرئيس من البرنامج. فيمكنك مثلاً، كتابة دالة ، وترسل لها طول ضلع مربع، فتقوم

بإرجاع مساحة ذلك المربع. والبرامج الضخمة هي عبارة عن مئات ، وربما آلاف من الدوال. وهناك دوال تقوم بأخذ قيمة وإرجاعها ، والبعض منها تقوم بأخذ قيمة ولاترجع قيمة إلى الجزء الرئيس من البرنامج، وهناك دوال لاتأخذ قيمة ولكن تقوم بإرجاع قيمة، وبعضها لاتقوم بأخذ أو إرجاع أية قيمة.

ويمكن للدالة أن تأخذ عدة قيم ، ومن أنواع مختلفة (مثل int, char, float, double...) ولكنها لاتستطيع إرجاع أكثر من قيمة واحدة. وتكتب الدوال عادة بعد الجزء الرئيس من البرنامج ، كما سنرى لاحقاً .

ب - كتابة دالة تقوم بأخذ قيمة وإرجاعها :

الصيغة العامة لكتابة دالة تقوم بأخذ قيمتين وإرجاع قيمة هي :

(اسم المتغير النوع، اسم المتغير النوع) اسم الدالة نوع القيمة التي سترجعها

```
{
.
.
return القيمة التي سترجعها الدالة;
}
```

مثال :

```
int add (char x , char y)
{
char r ;
r = x + y ;
return r ;
}
```

في المثال السابق ، كتبنا دالة تقوم بأخذ قيمتين، ثم تقوم بجمع القيمتين المرسلتين إليها، ووضع ناتج الجمع في المتغير "r" ثم تقوم بإرجاع قيمة المتغير "r" إلى الدالة التي قامت باستدعاء الدالة "add()".

ملاحظات حول الدالة السابقة:

1- قد تتساءل عزيزي القارىء، فتقول: لماذا لم نختصر الخطوة التي تخص تعريف المتغيرات التي ستتسلمها الدالة، أي كتابة (char x,y) طالما أن المتغيرين يتشابهان في النوع؟ والجواب هو أنه

لايجوز اختصار هذه الصيغة حتى ولو كان المتغيران متشابهين في النوع، ولكن يصح هذا الاختصار عند تعريف المتغيرات في الجزء الرئيس في البرنامج ، أو ضمن إحدى الدوال، ولكن لايجوز الاختصار في السطر الخاص بتعريف الدوال.

2- نلاحظ أن المتغير "r" هو متغير محلي، أي لايمكن استخدامه في الجزء الرئيس من البرنامج ، أو في أية دالة أخرى ؛ بمعنى أننا لانستطيع استخدامه سوى في هذه الدالة فقط، ولكن يجوز تعريف المتغير ذاته في دوال أخرى أو في الجزء الرئيس من البرنامج، ولكن سيكون لكل منها قيمتها الخاصة.

3- نلاحظ أنه لايشترط أن تكون القيم التي تتسلمها الدالة مشابهة لتلك التي ترجعها.

4- يجوز أن تتسلم الدالة عدة قيم من أنواع مختلفة، كما ذكرنا سابقاً.

5- يمكن اختصار الدالة السابقة بحيث نقوم بكتابة: (return x+y;) بدلاً مما كتبناه .

قلنا إن الدوال تكتب عادة بعد الجزء الرئيس للبرنامج، ويجب تعريف الدوال المستخدمة في البرنامج لهذا الجزء. ولتعريف الدالة السابقة التي كتبناها للجزء الرئيسي من البرنامج علينا أن نقوم بكتابة السطر الأول الخاص بتعريف الدالة كما هو، قبل الجزء الرئيس للبرنامج، ولكن نضيف في آخره الفاصلة المنقوطة، وكما يأتي:

```
int add (char x , char y);
```

البرنامج التالي، يقوم بإرسال نصف قطر دائرة إلى الدالة "circle" وتقوم هذه الدالة بإرجاع مساحة الدائرة:

```
#define pi 3.14

float circle (float r) ;

main ( )
{
float area ;
area = circle (5.0) ;
}

float circle (float r)
{
return r * pi ;
}
```

إن البرنامج السابق لن يقوم بعرض أية نتيجة على الشاشة لأننا لم نكتب له أمر الطباعة. وسنتعلم في الجزء الثاني من هذه السلسلة التعليمية بعون الله تعالى كيفية طباعة البيانات على الشاشة.

ج - كتابة دالة لاتقوم بأخذ أو إرجاع أية قيمة:

لكتابة دالة لاتقوم بأخذ قيمة، نكتب بين قوسين التعليمة (void) عند تعريف الدالة . ولكتابة دالة لاتقوم بإرجاع أية قيمة، نكتب قبل اسم الدالة، التعليمة (void) أيضاً. وعند كتابتهما معاً لاتقوم الدالة بأخذ أو إرجاع أية قيمة، إلى الجزء الرئيس من البرنامج.

البرنامج التالي يحتوي على دالة تقوم بخزن الأسكي كود للحرف "a" في متغير. لاحظ أن كل مايوضع بين العلامتين ' ' يعني أنك تريد الأسكي كود لما وضع بينهما.

```
void ASCII_a ( ) ;
main ( )
{
ASCII_a ( ) ;
}
void ASCII_a ( )
{
char ch ;
'ch' = a ;
}
```

لاحظ أن فتح القوس وإغلاقه يعني أن الدالة لاتأخذ أية قيمة، أي كأننا كتبنا التعليمية (void).

وهكذا عزيزي القارىء وصلنا إلى نهاية الجزء الأول من هذه السلسلة التعليمية ، التي تعلمك البرمجة باستخدام لغة C . وإذا كتب لهذه السلسلة التعليمية النجاح فإنني أعدك أن أقوم بكتابة سلسلة تعليمية أخرى لتعليم البرمجة باستخدام لغة C++. وبعدها باستخدام لغة VC++.ولكن كي تتعلم البرمجة باستخدام لغة Visual C++ يجب أن تكون لديك بعض الأساسيات البسيطة عن لغتي C و C++ ؛ وهذا ماستتعلمه من هذه السلسلة التعليمية.