تحليل عددي

خوارزميات وبرمجة

جزء البرمجة بلغة

C++

الدكتور هشام عبهري

البرمجة بلغة Turbo C++

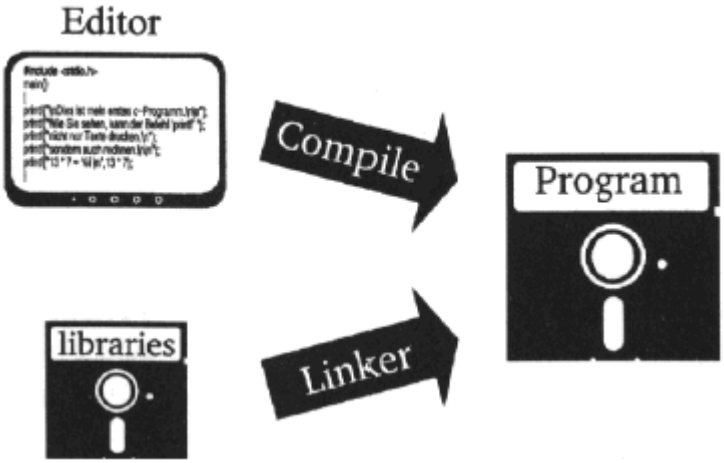
مقدمة:

تعتبر لغة الـ C++ تطورا للغة الـ C القياسية

كيف نكتب برنامجا بلغة C++

يكتب أي برنامجا بلغة C++ على ثلاثة مراحل:

1. كتابة نص البرنامج باستخدام أي محرر نصوص بحروف غير منسقة.
2. ترجمة هذا نص البرنامج إلى لغة الآلة وهذا ما يسمى بـ Compile.
3. ربط البرنامج بالمكتبات المنادى لها بهذا البرنامج وهذا ما يسمى بـ Link.



المكتبات Libraries

كون أن عدد التوابع التي تحتويها لغة الـ C++ وهذا ما يعتبر ميزة من ميزات هذه اللغة فإن العديد من التوابع قد هيأت ووضعت في مجلد يدعى المكتبات Libraries, حيث يستطيع المبرمج استخدام هذه التوابع ولا حاجة لبرمجتها مرة أخرى فمثلاً فهناك أكثر من مكتبة تدعم أوامر الإدخال والإخراج فمكتبة stdio تدعم أمر الإدخال scanf وأمر الإخراج printf أما مكتبة iostream تدعم أمر الإدخال cin وأمر الإخراج cout.
في هذا الكتاب سوف نعتمد المكتبة stdio لانجاز عمليتي الإدخال والإخراج, ولكننا سوف نوضح بشكل مقتضب عمليتي الإدخال والإخراج في المكتبة iostream.

البرنامج الصفري

يدعى البرنامج في المثال التالي بالبرنامج الصفري حيث يقرأ المترجم عبارة main() إشارة إلى البرنامج الرئيسي ثم إشارة } التي تدل على بداية البرنامج حيث يقوم بفتح نافذة الخرج ولكنه يغلقها مباشرة عند وصوله إلى الإشارة { ويعمل هذا البرنامج بدون أي أخطاء ولكن يقوم بتحذيرك بأن التابع main() يجب أن يقوم بإرجاع قيمة عن طريق الأمر return الذي سنشرحه فيما بعد (يمكنك تجاوز هذا التحذير بكتابة void قبل عبارة main()).

```
main()
{
}
```

التعليقات:

أثناء كتابة البرنامج تستطيع إدراج بعض التعليقات للتوضيح وهناك نوعان من التعليقات النوع الأول يبدأ بالقوس*/ وينتهي بالقوس /* ويمكننا كتابة ما نشاء بعده أما النوع الثاني فيأخذ سطراً كاملاً حيث يدل عليه وجود الإشارة // في أول السطر حيث البرنامج التالي يقوم بنفس عمل البرنامج السابق تماماً:

```
/* main program */
main()
{
}
// the end
```

الخرج على الشاشة printf :

هناك عدة أوامر للإخراج على الشاشة ويعد printf من الأوامر الرئيسية لذلك وصيغته العامة لإخراج النصوص هي:

```
printf("النص المراد إخراجه");
```

ويمكننا إضافة الرمز \n في أي مكان في النص حيث يقوم المترجم عند الإخراج بنقل المؤشر إلى بداية السطر التالي ومن الجدير بالذكر هنا أيضا انه يمكننا استخدام الرمز \t للانتقال ضمن السطر الواحد من مكان مؤشر الكتابة نحو اليمين بمقدار "Tab" أي ما يساوي ثمانية خانات ويمكننا استخدام الرمز \a لإصدار إنذار صوتي عند الوصول إلى تنفيذ الأمر المضاف إليه هذا الرمز:

وفي البرنامج التالي يستخدم الأمر printf لطباعة النص ! Hello ثم الانتقال إلى السطر التالي:

```
/* Example 1 */
#include<conio.h>
#include<stdio.h>
main()
{
printf( "Hello !\n" );
getch();
return 0;
}
```

يستخدم الأمر getch() في المترجمات التي تعمل في البيئة النصية لإيقاف مؤقت لشاشة الخرج حتى ضغط أي مفتاح ولا حاجة لاستخدام هذا الأمر في المترجمات التي تعمل في البيئة الرسمية.

أما Include فهو يستخدم لاستدعاء المكتبات حيث الملف stdio.h يحتوي على المكتبة التي تسمى بـ stream والتي تحوي معظم الأوامر الأساسية
أما الملف conio.h فيحتوي على المكتبة التي تسمى بـ console والتي استدعيناها من أجل الأمر getch(), ويرمز الحرفان io إلى Input/Output

أما الصيغة الثانية للأمر Printf فتستخدم لطباعة المتحولات Variables والصيغ الجبرية والصيغة العامة:

```
printf("%<output type>",<variable or expression>);
```

ويضم الجدول التالي بعض أنواع الصيغ الجبرية:

حرف واحد Character	**c**
عدد فاصلة عائمة float	**f**
عدد صحيح integer	**i**
سلسلة محرفية string	**s**
عدد فاصلة عائمة دقيق double	**lf**

وفي البرنامج التالي يقوم بطباعة نتيجة ضرب 3*2 على أنها عدد صحيح حيث يقوم بإخراج 6 أما إذا استبدلنا %i بـ %f سيصبح الناتج: 6.000000.

```
/* Example 2 */
#include<conio.h>
#include<stdio.h>
main()
{
printf("%i",2*3);
getch();
return 0;
}
```

المتحولات Variables

المتحول هو عبارة عن حجرة فارغة في الذاكرة جاهزة لوضع قيمة معينة فيها وتأخذ مساحة معينة حسب نوع القيم التي سنعطيها لها و الصيغة العامة للتصريح:

```
int x,y,z;
main()
{
.
.

}
```

ويبين الجدول التالي بعض أنواع المتحولات وحجمها من الذاكرة:

نوع البيانات	الوصف	الحجم من الذاكرة
Int	عدد صحيح	2 بايت
Short	عدد صحيح قصير	2 بايت
Long	عدد صحيح طويل	2 أو 4 بايت (تختلف من حاسب لآخر)
unsigned	عدد صحيح موجب	2 بايت
Char	حرف واحد	1 بايت
Signed char	حرف واحد يحتوي قيمة عددية من 128- حتى +127	1 بايت
Unsigned char	حرف واحد يحتوي قيمة عددية من 0 حتى 255	1 بايت
Float	عدد حقيقي يحتوي فاصلة عائمة	2 بايت
Double	عدد حقيقي يحتوي فاصلة عائمة وهو أدق منfloat	4 بايت
Long double	عدد حقيقي طويل يحتوي فاصلة	4 بايت أو أكثر تختلف من حاسب لآخر

ولمعرفة الحجم الذي يشغله متحول ما من الذاكرة نستخدم الأمر sizeof كما في المثال التالي:

```
/* Example 3 */
#include <stdio.h>
float x;
main()
{
x=2*3;
printf( "%f" ,x);
printf( "\n%i" ,sizeof(x));
return 0;
}
```

الثوابت Constant:

الثابت هو عبارة عن قيمة تمثل باسم لا نستطيع تغيير قمتها حيث يقوم المترجم باستبدال الاسم بالقيمة المسندة إليه والصيغة العامة لتعريفها هي كما يلي:

```
const <Constant name> = <Value>;
```

فمثلاً يقوم الأمر التالي بإسناد القيمة 3 إلى الثابت x

```
const x = 1+2;
```

تعليمة الإدخال العامة scanf :

Scanf من تعليمات الإدخال الرئيسية حيث تقوم بقراءة الحرف أو العدد المراد إدخاله ثم الضغط على enter للإشارة إلى انتهاء الإدخال والصيغة العامة لها:

```
scanf("%<input type>",&<variable>);
```

وفي المثال التالي يقوم البرنامج بحساب ناتج ضرب عددين:

```
/* Example 4 */
#include<conio.h>
#include<stdio.h>
int x,y,z;
main()
{
scanf("%i",&x);
scanf("%i",&y);
z=x*y;
printf("%i * %i = %i",x,y,z);
getch();
return 0;
}
```

وإذا أردنا تحديد دقة الخرج فإننا نضع نقطة قبل إشارة % ثم نكتب عدد الأرقام بعد الفاصلة كما يلي:

```
Printf( "%.عدد الأرقام بعد الفاصلةf" );
```

والمثال التالي يقوم بقراءة عددين حقيقيين ويقوم بطباعة الخرج على مرحلتين الثانية أكثر دقة من الأولى

```
/* Example 5 */
#include<conio.h>
#include<stdio.h>
float  x,y,z;
double c;
main()
{
scanf("%f",&x);
scanf("%f",&y);
z=x/y;
c=x/y;
printf("%f / %f = %.34f\n",x,y,z);
printf("%f / %f = %.34lf",x,y,c);
getch();
return 0;
}
```

المعاملات والصيغ الجبرية:

يوضح الجدول التالي العمليات الحسابية الرئيسية في الـ Turbo C++

العملية	الغرض منها
+	الجمع
-	الطرح
*	الضرب
/	القسمة
%	باقي القسمة

أمثلة على المعاملات السابقة:

لنفرض أن a و b عددان صحيحان حيث يأخذان القيمتان 7 و 3 على التوالي فيما يلي جدول بالقيم التي تنتج عن المعاملات المطبقة عليهما:

الصيغة الجبرية	الناتج
a + b	9
a - b	5
a * b	14
a / b	3
a % b	1

لنفرض كما في المثال السابق أن y و x عددان حقيقيان حيث يأخذان القيمتان 3.0 و 18.75 على التوالي :

الصيغة الجبرية	الناتج
x + y	21.75
x – y	15.75
x * y	56.25
x / y	6.25

لنفرض كما في المثالان السابقان أن i و j حرفان من النوع Character حيث يمثلان الحرفين A وG على التوالي:

الصيغة الجبرية	الناتج
i	65
i + j	136
i + j + 3	139
i + j + '3'	187

لنفرض أن a = 11 و b = 7.5 و c = 'x' فيما يلي جدول بالقيم التي تنتج عن المعاملات المطبقة عليهم :

الصيغة الجبرية	الناتج	نوعه
a + b	18.5	double حقيقي
a + c	99	integer صحيح
a + c + '2'	149	integer صحيح
(a + c) – (2 * b / 5)	96	integer صحيح

```
/* Example 6 */
#include <stdio.h>
int x,y;
main()
{
/* 1 */ x = 2 * 2;
/* 2 */ x = x++;
/* 3 */ x = ++x;
y = 4;
printf( "%i" ,x%y);
return 0;
}
```

يقوم البرنامج السابق في المرحلة الأولى بإسناد القيمة 2 * 2 أي 4 إلى المتحول الصحيح x وفي المرحلة الثانية يقوم بزيادته عدداً واحداً ويفعل الشيء نفسه في المرحلة الثالثة حيث تصبح قيمته أخيراً 6 ثم يقوم بطباعة باقي قسمة x/y وهي 2.

المعاملات المنطقية:

المعاملات المنطقية هي المعاملات التي تعطي True أو False عند تطبيقها على صيغتين وهي:

أصغر من	<
أصغر أو يساوي	<=
أكبر من	>
أكبر أو يساوي	>=
يساوي	==
لا يساوي	!=
and	&&
Or	||
Not	!

مثال:

a = 2 , b = 4 , c = 6

الصيغة	النتيجة	القيمة
a < b	True	1
(a + b) >= c	True	1
(b + c) < (a + 5)	False	0
c != 6	False	0
a == 2	True	1
(a = 2) && (a < 3)	True	1
(a = 2) || (a > 100)	True	1
!(a == 2)	False	0

أولوية المعاملات الجبرية

العمليات	فئة المعامل
sizeof ! -- ++ -	
% / *	الضرب و الجمع و باقي القسمة
- +	جمع و طرح (رياضيات)
=< < => > <	معاملات الربط
=! ==	التساوي
&&	And المنطقية
||	Or المنطقية
? :	المعاملات الشرطية
=% =/ =* =- =+ =	معاملات الإلحاق

حيث a += 2 تعادل a = a + 2

أما المعاملات الشرطية فهي كالتالي:

```
x = ( a < b ) ? a : c;
```

حيث تعني تلك العبارة أنه إذا كانت a < b فإن x = a وإلا فستصبح x = c

اذهب إلى Goto :

يقوم هذا الأمر بالتنقل بين أسطر البرنامج وشكله العام:

Goto < identifier > ;

فمثلاً هذا البرنامج يسألك إن كنت تريد المتابعة فإذا أجبت بـ y فإنه يطبع Welcome وإذا أجبت بـ n فهو يطبع good bye و إلا فإنه يعيد السؤال

```
/* Example 7 */
#include<conio.h>
#include<stdio.h>
char i;
main()
{
printf( "\n Continue (Y/N)? " );
question:
i = getch();
switch(i)
{
case 'y' : printf( "Welcome" ); break;
case 'n' : printf( "Good Bye" ); break;
default  : goto question;
}
}
```

If الشرطية:

تستخدم عبارة If لفحص الشرط فإذا تحقق أي كانت قيمته True فإنه ينفذ مجموعة أوامر أما إذا لم يتحقق أي كانت قيمته False فإنه ينفذ أوامر أخرى وله صيغتان وهما:

1. If (<condition>) {
   <group of statements>
   }

2. If (<condition>) {
   <group of statements 1>
   }
   Else {
   <group of statements 2>
   }

وعند وجود أمر واحد فقط للتنفيذ عندها يمكن الاستغناء عن القوسين والمثال التالي يوضح مبدأ عمل If

```
/* Example 8 */
#include<stdio.h>
char z;
main()
{
scanf("%s",&z);
if(z=='a' || z=='A' && z!='x')
    {
        printf("I am Addition");
    }
else
if(z=='s' || z=='S')
    {
        printf( "I am Subtraction" );
    }
else
if(z=='m' || z=='M')
    {
        printf( "I am Multiplication" );
    }
else
if(z=='d' || z=='D')
    {
        printf( "I am Division" );
    }
else
    {
        printf( "I Can't do it" );
    }
return 0;
}
```

عبارة Case:

تستخدم عبارة Case لفحص أكثر من شرط واحد والصيغة العامة للأمر case هي:

```
switch ( <switch variable> )
 {
case <constant expression1>  :
case <constant expression2>  :
case <constant expression3>  :
{
        <group of statements>
}
 break;
.
.
.
.
default    :
 {
       <group of statements>
}
}
```

وتقابل عبارة default في الأمر case العبارة else في الأمر if

فمثلاً البرنامج التالي يعمل نفس عمل البرنامج 7 ولكن بدقة أكبر وسهولة أكثر

```
/* Example 9 */
#include<stdio.h>
#include<conio.h>
char z;
main()
{
z = getch();
switch (z) {
case '+' :
case 'a' :
case 'A' :  {
         printf("I am Addition");
      }  break;
case '-' :
case 's' :
case 'S' :  {
   printf( "I am Subtraction" );
}  break;
case '*' :
case 'm' :
case 'M' :  {
   printf( "I am Multiplication" );
}  break;
case '/' :
case 'd' :
case 'D' :  {
       printf( "I am Division" );
      }  break;
case '=' :
case 'e' :
case 'E' : break;
default  : printf("I Can't do it");
}
return 0;
}
```

الحلقات التكرارية

الحلقات التكرارية تقوم بتكرار مجموعة من الأوامر لعدد معين من المرات

- حلقة For

وتقوم بتكرار عدد معين من الأوامر وتتميز بأنها تحتوي على متحول تتغير قيمته عند كل دورة فمثلاً البرنامج التالي يقوم بطباعة الأعداد من 1 إلى 10 حيث تزداد قيمة i في كل دورة عدداً واحداً

```
/* Example 10 */
#include<conio.h>
#include<stdio.h>
int i;
main()
{
for(i=1;i<=10;i++)
    {
        printf("%i\n",i);
    }
getch();
return 0;
}
```

أما البرنامج التالي فيطبع الأعداد من 1 إلى 20 حيث يزداد i ثلاثة في كل دورة

```
/* Example 11 */
#include<conio.h>
#include<stdio.h>
int i;
main()
{
for(i=1;i<=20;i=i+3)
    {
        printf("%i\n",i);
    }
getch();
return 0;
}
```

أما البرنامج التالي فيقوم بطباعة مجموع الأعداد من 1 إلى 10

لاحظ أنه قد قمنا بإعطاء قيمة مبدئية ل s وهي الصفر أما إذا أردنا إظهار ناتج ضرب الأعداد من 1 إلى 10 فإننا نستبدل إشارة الجمع بإشارة الضرب و نهيئ s بالقيمة 1.

```
/* Example 12 */
#include<conio.h>
#include<stdio.h>
int i,s;
main()
{
s=0;
for(i=1;i<=10;i++)
    {
        s=s+i;
    }
   printf("%i\n",s);
getch();
return 0;
}
```

- حلقة While

تقوم حلقة while بتنفيذ عدد من الأوامر مادام الشرط محقق أي أنه لا يوجد عدد معين للتكرار فمثلاً البرنامج التالي يقوم بقراءة قيمة z فإذا كانت تساوي 'y' فإنه يطبع عبارة I am here ثم يعيد قراءة z حتى ينكسر الشرط أي تصبح z لا تساوي 'y' فيقوم بالخروج من الحلقة وينفذ الأمر getch().

```
/* Example 13 */
#include<conio.h>
#include<stdio.h>
char z;
main()
{
scanf ( "%s" ,&z);
while(z=='y')
    {
        printf("I am here\n");
        scanf ("%s",&z);
    }
getch();
return 0;
}
```

- حلقة Do-While

تتميز حلقة Do-While عن حلقة While بأنها تقوم بتنفيذ الأوامر أولاً ثم تفحص الشرط ثانياً أي أن البرنامج التالي يقوم بطباعة عبارة I am here ثم يقوم بقراءة z فإذا كانت لا تساوي Y فإنه يخرج من الحلقة مباشرة .

```
/* Example 14 */
#include<stdio.h>
char z;
main()
{
do
   {
      printf("I am here\n");
     scanf ("%s",&z);
   }
while(z=='y');
return 0;
}
```

التوابع Functions:

التابع هو عبارة عن مجموعة من الأوامر موضوعة تحت اسم معين ويتم استدعاءها أثناء البرنامج الأساسي أكثر من مرة بذكر اسم التابع والصيغة العامة له:

```
<Function Type> <Function name> ( <parameters> )
{
<group of statements>
return <value>;
}
```

نوع التابع هو نوع الصيغ الناتجة عن التابع

أما الوسائط فهي عبارة عن متحولات نقوم بإعطائها للتابع ليقوم بمعالجتها وإعطاء قيمة في الأمر return.

فمثلاً البرنامج التالي يقوم بإعطاء قيمة y إلى التابع add حيث يقوم التابع باستبدال x بقيمة y ويزيدها بمقدار 1 ويطبعها

تماماً كما في المعادلات الجبرية ولكنه فعلياً لا يزيد قيمة y لأنه في الأمر return يعيد القيمة 0.

```
/* Example 15 */
#include <stdio.h>
int y;
int add(int x)
{
x = x+1;
printf( "\n%i" ,x);
return 0;
}

main()
{
y = 2;
printf( "%i" ,y);
add(y);
printf( "\n%i" ,y);
return 0;
}
```

```
/* Example 16 */
#include <stdio.h>

int y;
int add(int x)
{
x = x + 1;
printf( "\n%i" ,x);
return x;
}

main()
{
y = 2;
printf( "%i" ,y);
y = add(y);
printf( "\n%i" ,y);
return 0;
}
```

ولفهم التوابع بشكل أدق لنلاحظ في البرنامج السابق بأنه تم إسناد التابع add(y) إلى القيمة y فهذا يعني أن المترجم يقوم بإسناد قيمة التابع إلى المتحول y فهذا يعني أن التابع له قيمة وهي تعطى بالأمر return أما إذا كانت 0 فيصبح عمل التابع كالإجراء void تماماً ولا يقوم بإرجاع أي قيمة وفي البرنامج التالي يقوم البرنامج الأساسي بإسناد القيمة 6 إلى المتحول y.

```
/* Example 17 */
#include <stdio.h>

int y;

int add(int z,int a)
{
int x;
x = z + a;
return x;
}

main()
{
y = add(4,2);
printf( "\n%i" ,y);
return 0;
}
```

وفيما يلي جدول ببعض التوابع القياسية في لغة Turbo C++

التابع	نوعه	الغرض منه	المكتبة التي تحويه
abs(x)	int	يعيد القيمة المطلقة لـ x	stdlib.h
cos(x)	double	يعيد تجب الزاوية x	math.h
log(x)	double	يعيد اللوغاريتم الطبيعي لـ x	math.h
sqrt(x)	double	يعيد الجذر التربيعي لـ x	math.h
sin(x)	double	يعيد جيب الزاوية x	math.h
tan(x)	double	يعيد ظل الزاوية x	math.h
tolower(x)	int	يحول الحرف إلى حرف صغير	stdlib.h or ctype.h
toupper(x)	int	يحول الحرف إلى حرف كبير	stdlib.h or ctype.h
toascii(x)	int	يحول الحرف إلى الأسكي	ctype.h
strlen(x)	int	يعيد طول السلسلة المحرفية x	string.h

التابع void :

إذا كان نوع التابع void فإنه لا يصلح أن نضع الأمر return أبداً أي أن النوع void هو تابع لا يعيد أي قيمة

فمثلاً البرنامج التالي يقوم بطباعة العبارة Hello ومن ثم يقوم بطباعة مجموع العددين 3 و 7 .

```
/* Example 18 */
#include <stdio.h>
void hello()
{
printf( "hello" );
}
void add(int x,int y)
{
int z;
z = x + y ;
printf( "\n%i" ,z);
}
main()
{
hello();
add(3,7);
return 0;
}
```

المصفوفات Arrays:

```
/* Example 19 */
#include <stdio.h>
int i,ray[10];
main()
{
i = 1;
for (i == 1 ;i <= 10;i++) scanf( "%i"
,&ray[i]);
i = 1;
for (i == 1 ;i <= 10;i++) printf( "\n%i"
,ray[i]);
return 0;
}
```

البرنامج التالي يقوم بتعريف البرنامج الجزئي read لقراءة مصفوفة ما, ثم تعريف البرنامج الجزئي wirte لطباعة مصفوفة ما.

```
/* Example 20 */
#include <stdio.h>

void read(int x[])
{
int i;
for (i=1;i<=10;i++)
{
scanf( "%i" ,&x[i]);
}
}
void write(int x[])
{
int i;
for (i=1;i<=10;i++)
{
printf( "%i" ,x[i]);
}
}
int y[10];
main()
{
read(y);
write(y);
return 0;
}
```

نلاحظ إننا استخدمنا في هذا البرنامج تقنية البرامج الجزئية بمتحول واحد حيث أننا عرفنا هنا التابعين read( ) والتابع wirte( ) والمتحول هنا مصفوفة ذات بعد واحد

في المثال 23 سوف نستخدم تقنية تعريف متحول #define لكتابة برنامج جداء مصفوفتين.

الماكرو MACROS

هو مجموعة من التعليمات تؤدى غرض معين ويشبه إلى حد كبير الدالة function ويتم إنشاء الماكرو مرة واحدة وبعد ذلك يمكن استدعائه كلما احتجت إليه (أي يتم تعريف الثوابت أو عمليات محدده في بداية البرنامج وتكون لها صفه العمومية للاستخدام داخل الدالة الرئيسية والدوال الفرعية)

كيفية إنشاء الماكرو:

يتم ذلك باستعمال الكلمة #define وهذه الكلمة تسمى directive أو preprocessor ومعناها التوجيه

الصورة العامة:

```
#define macro line
```

مثلا

```
#define a 5
```

وهى عبارة عن تعريف طرف بطرف, ومعناه عرف المتحول a بالقيمة 5

إن المثال التالي يوضح كيفية الإعلان عن الماكرو وكيفية استعماله

```
/* Example 21 */
# include<stdio.h>
# define sum(a,b) a+b
# define mul(x,y) x*y
main ( )
{
int v1=5 , v2 = 10;
printf("\n\n sum(v1,v2) = % d",sum(v1,v2));
printf("\n\n mul(v1,v2) = % d",mul(v1,v2));
return 0;
}
```

```
/*  Example 22 */
#include<stdio.h>
#define title "\n\t\t\t فـاتـورة\n\n\n"
#define pp1 5        سعر الـقـطـعـة الاولـى//
#define pp2 10      سعر الـقـطـعـة الـثـانـيـه//
#define n1 4       عدد الـقـطـع الـمـشـتـراة مـن الـصـنـف الاول//
#define n2 8      عدد الـقـطـع الـمـشـتـراة مـن الـصـنـف الـثـانـي//
int p1,p2;
main()
{
printf(title);
p1=pp1*n1;        الـسـعـر الـكـلـي الـلاول//
p2=pp2*n2;       الـسـعـر الـكـلـي الـثـانـي//
printf("\n سعر الـقـطـعـة        عدد الـقـطـع        الـسـعـر الـكـلـي");
printf("\n_____________________________________");
printf("\n   %i",p1);
printf("                          %i",pp1);
printf("                         %i",n1);

printf("\n\n   %i",p2);
printf("                          %i",pp2);
printf("                         %i",n2);
```

```
printf("\n\n_____________________________");
printf("\n\n  %i",p1+p2);
printf("              السعر الإجمالي          ");
return 0;
}
```

إذا نفذنا المثال 22 لحصلنا على الناتج التالي:

فاتورة

سعر القطعة	عدد القطع	السعر الكلي
4	5	20
8	10	80
	السعر الاجمالي	100

وسنرى فيما بعد أننا سوف نحصل على هذه النتيجة بطرق عديدة.

البرنامج التالي يحسب المصفوفة الناتجة عن جداء مصفوفتين

```
/*  Example 23 */
#include<stdio.h>
#define l 4
#define n 2
#define m 3
int i,j,k,a[n][l],b[l][m],c[n][m];
main()
{
for(i=1;i<=n;i++)

for(k=1;k<=l;k++)
     scanf("%i",&a[i][k]);
for(k=1;k<=l;k++)
for(j=1;j<=m;j++)
     scanf("%i",&b[k][j]);
for(i=1;i<=n;i++)
         for(j=1;j<=m;j++)
              {
               c[i][j]=0;
                    for(k=1;k<=l;k++)

c[i][j]+=a[i][k]*b[k][j];
              }
printf("\n");
for(i=1;i<=n;i++)
{
printf("\n");
for(j=1;j<=m;j++)
     printf("%i      ",c[i][j]);
 }
return 0;
}
```

المؤشرات Pointer

لقد تعملنا حتى الآن مع المتحولات التي تأخذ قيمة لا تتغير طيلة تشغيل البرنامج وبالتالي فان هذا المتحول يحجز في الذاكرة عنوان معين طيلة تشغيل البرنامج .

تعتبر المؤشرات من أهم الخصائص التي تمتاز بها لغة C++ وتستخدم في البرامج التي تحتاج فيها إلى حجز وإلغاء الذاكر أثناء تنفيذ البرنامج.

مهمة المؤشر **pointer** هو التعامل مع محتوى العناوين في الذاكرة

الملفات Folder

فتح الملفات وإغلاقها

قراءة الملفات

الكتابة في الملفات

طباعة الملفات

حذف الملفات

المقاطعات Interrupt

مقاطعات الحاسب الشخصي

توابع الـ INT 10

إحداثيات مؤشر الكتابة

تنظيف الشاشة

لف (تدوير) الشاشة Rollen

التراكيب Struc,Union ,Tybe

تطبيق لغة الـ C++ في مجالات مختلفة

سوف نورد في هذا الفصل حلولا لبعض الأمور العلمية التي تواجه الباحث في حياته اليومية سواء أكانت من علم الرياضيات أو التجارة أو ...الخ.

تطبيق 1

إيجاد حلول المعادلات الخطية من المرتبة الثانية

تطبيق 2

إيجاد حلول المعادلات الخطية من المرتبة الثالثة

تطبيق 3

إيجاد حلول المعادلات الخطية من المرتبة الرابعة

تطبيق 4

إيجاد حاصل جداء مصفوفتين

من العمليات على المصفوفات المستطيلة, هي عملية ضرب المصفوفات.
لتكن لدينا المصفوفة A والمصفوفة B ولتكن المصفوفة C ناتج الجداء أي:
$C = A * B$.

شرط جداء مصفوفتين

فإذا كانت لدينا المصفوفة $(A)_{n.p}$ المؤلفة من n سطراً و p عموداً و المصفوفة $(B)_{q.m}$ المؤلفة من q سطراً و m عموداً و علمنا إن شرط وجود المصفوفة C ناتج الجداء هو أن يكون عدد أعمدة المصفوفة الأولى A مساويا لعدد اسطر المصفوفة الثانية B أي يجب أن يكون

$$p = q = l$$

فإن المصفوفة عدد اسطر المصفوفة الناتجة C يساوي عدد اسطر المصفوفة الأولى A و عدد أعمدتها مسويا عدد لعدد أعمدة المصفوفة الثانية B أي يجب أن يكون:

$$(A)_{nl} * (B)_{lm} = (C)_{nm}$$

آلية جداء مصفوفتين

ان العنصر $c_{ij} \in (C)_{nm}$ هو ناتج جداء السطر i من المصفوفة A بـ العمود j من المصفوفة B , ويتم هذا الجداء على الشكل التالي:

$$\begin{array}{c} \\ \\ \\ \\ \\ \text{السطر i} \rightarrow \\ \\ \\ \\ \end{array} \begin{pmatrix} a_{11} & a_{12} & . & . & . & a_{1l} \\ a_{21} & a_{22} & . & . & . & a_{2l} \\ . & & & & & \\ . & & & & & \\ . & & & & & \\ a_{i1} & a_{i2} & . & . & . & a_{il} \\ . & & & & & \\ . & & & & & \\ . & & & & & \\ a_{n1} & a_{n2} & . & . & . & a_{nl} \end{pmatrix} * \begin{array}{c} \text{العمود j} \downarrow \\ \begin{pmatrix} b_{11} & b_{12} & . & . & . & b_{1j} & . & . & . & b_{1m} \\ b_{21} & b_{22} & . & . & . & b_{2j} & . & . & . & b_{2m} \\ . & & & & & & & & & \\ . & & & & & & & & & \\ . & & & & & & & & & \\ b_{l1} & b_{l2} & . & . & . & b_{lj} & . & . & . & b_{lm} \end{pmatrix} \end{array}$$

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + ... + a_{il}b_{lj}$$

ويمكن كتابة هذه العلاقة بالشكل التالي:

$$c_{ij} = \sum_{k=1}^{l} a_{ik} b_{kj} \qquad ; i = 1(1)n \quad \& \quad j = 1(1)m$$

مثال

أوجد ناتج الجداء المصفوفي التالي:

$$C = A * B$$

حيث أن

$$A = \begin{pmatrix} 1 & 2 & -1 \\ 0 & 1 & 3 \\ -2 & 2 & 0 \\ 1 & 2 & 3 \end{pmatrix} \qquad ; \qquad B = \begin{pmatrix} 2 & 0 & 1 & 2 & 3 \\ -1 & 1 & 2 & 1 & 0 \\ 2 & _1 & 3 & 2 & 1 \end{pmatrix}$$

باستخدام الآلية السابقة نجد:

$$C = \begin{pmatrix} -2 & 3 & 2 & 2 & 2 \\ 5 & -2 & 11 & 7 & 3 \\ -6 & 2 & 2 & -2 & -6 \\ 6 & -1 & 14 & 10 & 6 \end{pmatrix}$$

البرنامج التالي هو برنامج يعطي ناتج جداء مصفوفتين من نوع double

```
/*  Example 24 */
#include<stdio.h>
#include<conio.h>
#define n 4
#define l 3
#define m 5
int i,j,k;
double a[n][l],b[l][m],c[n][m];
main()
{
for(i=1;i<=n;i++)
for(k=1;k<=l;k++)
     scanf("%lf",&a[i][k]);
for(k=1;k<=l;k++)
for(j=1;j<=m;j++)
     scanf("%lf",&b[k][j]);
for(i=1;i<=n;i++)
         for(j=1;j<=m;j++)
              {
               c[i][j]=0;
                    for(k=1;k<=l;k++)

c[i][j]+=a[i][k]*b[k][j];
              }
clrscr();
printf("\n\na(%i,%i) ÇáãÕÝæÝÉ\n",n,l);
for(i=1;i<=n;i++)
{
printf("\n");
for(k=1;k<=l;k++)
     printf("%.1lf\t",a[i][k]);
}
printf("\n\nb(%i,%i) ÇáãÕÝæÝÉ\n",l,m);
for(k=1;k<=l;k++)
{
printf("\n");
```

```
for(j=1;j<=m;j++)
     printf("%.1lf\t",b[k][j]);
}
printf("\n\nc(%i,%i) ÇáãÕÝæÝÉ\n",n,m);
for(i=1;i<=n;i++)
{
printf("\n");
for(j=1;j<=m;j++)
     printf("%.1lf\t",c[i][j]);
 }
return 0;
}
```

لقد استخدمنا البرنامج السابق لإيجاد المصفوفة C ناتج جداء المصفوفتين A و B الواردتين في المثال الأخير وكانت نتائج هذا البرنامج كما يلي:

```
a(4,3) المصفوفة

1.0     2.0     -1.0
0.0     1.0     3.0
-2.0    2.0     0.0
1.0     2.0     3.0

b(3,5) المصفوفة

2.0     0.0     1.0     2.0     3.0
-1.0    1.0     2.0     1.0     0.0
2.0     -1.0    3.0     2.0     1.0

c(4,5) المصفوفة

-2.0    3.0     2.0     2.0     2.0
5.0     -2.0    11.0    7.0     3.0
-6.0    2.0     2.0     -2.0    -6.0
6.0     -1.0    14.0    10.0    6.0
```

وهذا ما يتطابق مع الناتج المحسوب يدوياً.

تطبيق 5

إيجاد الحل المشترك لجملة معادلات خطية من الرتبة n باستخدام طريقة غوص Gauß

لتكن لدينا جملة المعادلات الخطية التالية المؤلفة من n معادلة بـ n مجهول:

$$\left.\begin{array}{l} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{array}\right\} \qquad (I)$$

بالاعتماد على مبدأ ضرب و تساوي المصفوفات يمكننا كتابة جملة المعادلات السابقة بالشكل المصفوفي التالي:

$$\begin{pmatrix} a_{11} & a_{12} & . & . & . & a_{1n} \\ a_{21} & a_{22} & . & . & . & a_{2n} \\ . & & & & & \\ . & & & & & \\ . & & & & & \\ a_{n1} & a_{n2} & . & . & . & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ . \\ . \\ . \\ b_n \end{pmatrix} \qquad (II)$$

أي أن الجملة السابقة يمكن كتابتها بالشكل التالي:

$$A \quad X = B$$

حيث أن:

$$A \in \Re^n X \Re^n$$

$$X, B \in \Re^n X \Re^1$$

نسمي المصفوفة A بمصفوفة الأمثال ونسمي المصفوفة B عمود الطرف الأيمن للجملة ونسمي المصفوفة X بعمود المجاهيل.

نعلم أنه في جمل المعادلات الخطية تتميز بالخواص التالية:

- "إن ضرب إحدى المعادلات بعدد حقيقي ما وجمعها إلى معادلة أخرى من المعادلات لا يغير من حل هذه الجملة"
- "إن تبديل موقعي معادلتين في الجملة لا يغير من حل هذه الجملة"

تعتمد طريقة غوص Gauß

أ - على إنشاء المصفوفة المستطيلة المؤلفة من مصفوفة الأمثال موسعة بعمود الطرف الأيمن B, تسمى مصفوفة الأمثال الموسعة.ويكون شكل هذه المصفوفة على الشكل:

$$\left(\begin{array}{cccccc|c} a_{11} & a_{12} & . & . & . & a_{1n} & b_1 \\ a_{21} & a_{22} & . & . & . & a_{2n} & b_2 \\ . & & & & & & . \\ . & & & & & & . \\ . & & & & & & . \\ a_{n1} & a_{n2} & . & . & . & a_{nn} & b_n \end{array}\right) \qquad (III)$$

ولسهولة الحسابات سوف نجري الترميز التالي:

$$B = \begin{pmatrix} b_1 \\ b_2 \\ . \\ . \\ . \\ b_n \end{pmatrix} = \begin{pmatrix} a_{1,n+1} \\ a_{2,n+1} \\ . \\ . \\ . \\ a_{n,n+1} \end{pmatrix}$$

و عليه تصبح المصفوفة الموسعة على الشكل التالي:

$$\left(\begin{array}{cccccc|c} a_{11} & a_{12} & . & . & . & a_{1n} & a_{1,n+1} \\ a_{21} & a_{22} & . & . & . & a_{2n} & a_{2,n+1} \\ . & & & & & & . \\ . & & & & & & . \\ . & & & & & & . \\ a_{n1} & a_{n2} & . & . & . & a_{nn} & a_{n,n+1} \end{array}\right) \qquad (IV)$$

ب- جعل جميع الأمثال الواقعة تحت القطر الرئيسي في المصفوفة الموسعة (IV) أصفاراً, كما يلي:

- نضرب السطر الأول
  - o بـ $\left(-\frac{a_{21}}{a_{11}}\right)$ ونجمعه إلى السطر الثاني
  - o بـ $\left(-\frac{a_{31}}{a_{11}}\right)$ ونجمعه إلى السطر الثالث وهكذا .....
  - o بـ $\left(-\frac{a_{n1}}{a_{11}}\right)$ ونجمعه إلى السطر النوني

<u>والنتيجة</u>

جميع عناصر المصفوفة الموسعة (IV) قد تغيرت عدا عناصر السطر الأول وجميع العناصر الواقعة تحت العنصر a_{11} أصبحت أصفاراً.

- نضرب السطر الثاني
  - o بـ $\left(-\frac{a_{32}}{a_{22}}\right)$ ونجمعه إلى السطر الثالث
  - o بـ $\left(-\frac{a_{42}}{a_{22}}\right)$ ونجمعه إلى السطر الرابع وهكذا .....
  - o بـ $\left(-\frac{a_{n2}}{a_{22}}\right)$ ونجمعه إلى السطر النوني

<u>والنتيجة</u>

جميع عناصر المصفوفة الموسعة (IV) قد تغيرت عدا عناصر السطر الأول والثاني وجميع العناصر الواقعة تحت العنصر a_{22} أصبحت أصفاراً.

وهكذا حتى نصل للنتيجة أن جميع العناصر الواقعة تحت القطر الرئيسي في المصفوفة الموسعة (I) قد أصبحت أصفاراً.

ملاحظة:

نلاحظ مما سبق أن:

عناصر السطر الأول لم يجري عليها أي تغير.
عناصر السطر الثاني قد تغيرت مرة واحدة.
عناصر السطر الثالث قد تغيرت مرتين.
وهكذا فإن عناصر السطر النوني قد تغيرت $n-1$ مرت.

ثم نكتب الجملة (I) من جديد فنجد:

$$\left.\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= a_{1,n+1} \\ + a_{22}x_2 + \ldots + a_{2n}x_n &= a_{2,n+1} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{nn}x_n &= a_{n,n+1} \end{aligned}\right\} \qquad (V)$$

ت- إيجاد الحل x_n من المعادلة الأخيرة واعتباره حلاً ابتدائياً للحصول على الحل x_{n-1} من المعادلة ما قبل الأخيرة وهكذا حتى نصل إلى المعدلة الأولى حاملين الحلول x_n ، $x_{n-1},\ldots,x_2$ وبالتالي نحصل منها على الحل x_1 وفق العلاقات التالية:

$$x_n = \frac{1}{a_{nn}} a_{n,n+1}$$

$$x_i = \frac{1}{a_{ii}}\left[a_{i,n+1} - \sum_{k=i+1}^{n} a_{ik}x_k\right], \quad i = (n-1)(1)1$$

مثال

استخدم طريقة غوص في إيجاد حل جملة المعادلات الآتية:

$$\begin{array}{rcrcrcrcrcr} x_1 & + & 3x_2 & + & 2x_3 & - & x_4 & - & 2x_5 & = & 3 \\ -3x_1 & - & 8x_2 & - & 3x_3 & + & 2x_4 & + & 5x_5 & = & -7 \\ 4x_1 & + & 8x_2 & + & 5x_3 & - & 3x_4 & - & 4x_5 & = & 10 \\ -2x_1 & - & 4x_2 & - & x_3 & + & 2x_4 & + & 3x_5 & = & -2 \\ 5x_1 & + & 11x_2 & + & 7x_3 & - & x_4 & - & 7x_5 & = & 15 \end{array}$$

نكتب المصفوفة الموسعة:

$$A = \begin{pmatrix} 1 & 3 & 2 & -1 & -2 & | & 3 \\ -3 & -8 & -3 & 2 & 5 & | & -7 \\ 4 & 8 & 5 & -3 & -4 & | & 10 \\ -2 & -4 & -1 & 2 & 3 & | & -2 \\ 5 & 11 & 7 & -1 & -7 & | & 15 \end{pmatrix}$$

نضرب السطر الأول بـ:

(3) و نجمعه إلى السطر الثاني

(-4) ونجمعه إلى السطر الثالث

(2) ونجمعه إلى السطر الرابع

(-5) ونجمعه إلى السطر الخامس

فتصبح مصفوفة الأمثال على الشكل التالي:

$$A=\begin{pmatrix} 1 & 3 & 2 & -1 & -2 & | & 3 \\ 0 & 1 & 3 & -1 & -1 & | & 2 \\ 0 & -4 & -3 & 1 & 4 & | & -2 \\ 0 & 2 & 3 & 0 & -1 & | & 4 \\ 0 & -4 & -3 & 4 & 3 & | & 0 \end{pmatrix}$$

نضرب السطر الثاني بـ :

(4) ونجمعه إلى السطر الثالث

(-2) ونجمعه إلى السطر الرابع

(4) ونجمعه إلى السطر الخامس

فتصبح مصفوفة الأمثال على الشكل التالي:

$$A=\begin{pmatrix} 1 & 3 & 2 & -1 & -2 & | & 3 \\ 0 & 1 & 3 & -1 & -1 & | & 2 \\ 0 & 0 & 9 & -3 & 0 & | & 6 \\ 0 & 0 & -3 & 2 & 1 & | & 0 \\ 0 & 0 & 9 & 0 & -1 & | & 8 \end{pmatrix}$$

نضرب السطر الثالث بـ :

$\left(\frac{1}{3}\right)$ ونجمعه إلى السطر الرابع

(-1) ونجمعه إلى السطر الخامس

فتصبح مصفوفة الأمثال على الشكل التالي:

$$A=\begin{pmatrix} 1 & 3 & 2 & -1 & -2 & | & 3 \\ 0 & 1 & 3 & -1 & -1 & | & 2 \\ 0 & 0 & 9 & -3 & 0 & | & 6 \\ 0 & 0 & 0 & 1 & 1 & | & 2 \\ 0 & 0 & 0 & 3 & -1 & | & 2 \end{pmatrix}$$

نضرب السطر الرابع بـ:
(-3) ونجمعه إلى السطر الخامس

فتصبح مصفوفة الأمثال على الشكل التالي:

$$A=\begin{pmatrix} 1 & 3 & 2 & -1 & -2 & | & 3 \\ 0 & 1 & 3 & -1 & -1 & | & 2 \\ 0 & 0 & 9 & -3 & 0 & | & 6 \\ 0 & 0 & 0 & 1 & 1 & | & 2 \\ 0 & 0 & 0 & 0 & -4 & | & -4 \end{pmatrix}$$

نكتب جملة المعادلات الخطية من جديد فنجد:

$$\begin{array}{ccccccccccc} x_1 & + & 3x_2 & + & 2x_3 & - & x_4 & - & 2x_5 & = & 3 \\ & & x_2 & + & 3x_3 & - & x_4 & - & x_5 & = & 2 \\ & & & & 9x_3 & - & 3x_4 & & & = & 6 \\ & & & & & & x_4 & + & x_5 & = & 2 \\ & & & & & & & - & 4x_5 & = & -4 \end{array}$$

من المعادلة الأخيرة نجد أن :

$$\underline{\underline{x_5 = 1}}$$

من المعادلة الرابعة نجد أن:

$$x_4 = 2 - x_5 \quad \Rightarrow \quad \underline{\underline{x_4 = 1}}$$

من المعادلة الثالثة نجد:

$$9x_3 = 6 + 3x_4 \quad \Rightarrow \quad 9x_3 = 9 \quad \Rightarrow \quad \underline{\underline{x_3 = 1}}$$

من المعادلة الثانية نجد:

$$x_2 = 2 - 3x_3 + x_4 + x_5 \quad \Rightarrow \quad \underline{\underline{x_2 = 1}}$$

وأخيرا من المعادلة الأولى نجد:

$$x_1 = 3 - 3x_2 - 2x_3 + x_4 + 2x_5 \quad \Rightarrow \quad \underline{\underline{x_1 = 1}}$$

وبالتالي فان حل الجملة هو الشعاع التالي:

$$x = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

<u>مثال:</u>

اكتب برنامجا بلغة الـ C++ الذي يحسب تغيرات عناصر المصفوفة الموسعة لجملة المعادلات الخطية (I) بعد ضرب السطر الأول بالقيمة $\left(-\frac{a_{i1}}{a_{11}}\right)$ وجمعها للسطر i, حيث أن $i = 1(1)n$.

مثال:

اكتب برنامجا بلغة الـ C++ الذي يحسب تغيرات عناصر المصفوفة الموسعة لجملة المعادلات الخطية (I) بعد ضرب السطر الأول بالقيمة $\left(-\frac{a_{i1}}{a_{11}}\right)$ وجمعها للسطر i, حيث أن $i = 2(1)n$ ثم ضرب عناصر السطر الثاني بالقيمة $\left(-\frac{a_{i2}}{a_{22}}\right)$ وجمعها للسطر i حيث أن $i = 3(1)n$.

```
#include<stdio.h>
#define n 5
double s,a[10][10],x[10];
int i,j,k;
main()
{
for(i=1;i<=5;i++)
for(j=1;j<=6;j++)
scanf("%lf",&a[i][j]);

for(i=2;i<=n;i++)
{
        s=a[i][1]/a[1][1];
                for(j=1;j<=n+1;j++)
                        a[i][j]=a[i][j]-a[1][j]*s;
}

        for(i=1;i<=n;i++)
{
        printf("\n");
                for(j=1;j<=n+1;j++)
                        printf("%.2lf      ",a[i][j]);
}
return 0;
}
```

مثال:

أكتب برنامجا بلغة الـ ++C الذي يجعل جميع العناصر الواقعة تحت القطر الرئيسي في مصفوفة الأمثال الموسعة (I) أصفاراً.

```
#include<stdio.h>
#define n 5
double s,a[10][10],x[10];
int i,j,k;
main()
{
for(i=1;i<=5;i++)
for(j=1;j<=6;j++)
scanf("%lf",&a[i][j]);



for(k=1;k<=n-1;k++)
for(i=k+1;i<=n;i++)
{
        s=a[i][k]/a[k][k];
                for(j=1;j<=n+1;j++)
                        a[i][j]=a[i][j]-a[k][j]*s;
}



        for(i=1;i<=n;i++)
{
        printf("\n");
                for(j=1;j<=n+1;j++)
                        printf("%.2lf      ",a[i][j]);
}
return 0;
}
```

مثال:

اكتب برنامجاً بلغة الـ ++C والذي يعطي حل جملة المعادلات الخطية (I) بطريقة غوص.

تطبيق 6

إيجاد حل جمل المعادلات الخطية ذات الأطراف المتعددة.

تطبيق 7

استخدام طريقة غوص Gauß في إيجاد مقلوب المصفوفات المربعة.

تطبيق 8

إيجاد أكبر عدد من مجموعة ثلاثة أعداد .

سنفرض A و B و C ثلاثة أعداد معطاة, إن خوارزمية إيجاد أكبرها هي:

- إذا كان: $A > B$ عندئذ : إذا كان $A > C$ $\Leftarrow$ $Max = A$
وإذا كان $A \leq C$ $\Leftarrow$ $Max = C$
- إذا كان : $A \leq B$ عندئذ : إذا كان $B > C$ $\Leftarrow$ $Max = B$
وإذا كان $B \leq C$ $\Leftarrow$ $Max = C$

```
#include<stdio.h>
double a,b,c,max;
main()
{
scanf("%lf",&a);
scanf("%lf",&b);
scanf("%lf",&c);
if(a>b && a>c)
printf("Max =%lf",a);
else
if(a>b && a<=c)
printf("Max =%lf",c);
else
if(a<=b && b>c)
printf("Max =%lf",b);
else
if(a<=b && b<=c)
printf("Max =%lf",c);
return 0;
}
```

تطبيق 8

إيجاد أكبر عدد من مجموعة " n "عدداً

ليكن لدينا الأعداد التالية $x_1, x_2, ..., x_n$; $n \geq 2$ إن خوارزمية إيجاد أكبر عدد في هذه المجموعة هي:

أ - نفرض أن أكبر رقم هو x_1 (أي نضع $Max = x_1$).

ب - من أجل $i = 1(1)n$ نسأل هل $Max \geq x_i$, فإذا كان الجواب "لا" نضع $Max = x_i$.

```
#include<stdio.h>
int i,max,x[10];
main()
{
for(i=1;i<=5;i++)
scanf("%i",&x[i]);

max=x[1];
for(i=2;i<=5;i++)
        {
                if(max<x[i])
                        max=x[i];
        }
printf("%i",max);

return 0;
}
```

تطبيق 9

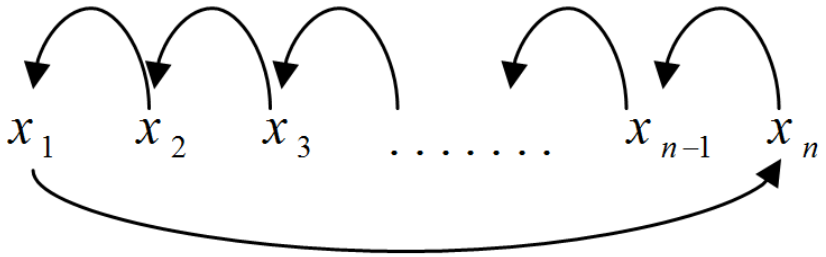
مبدأ النقل الدائري

–النقل موقعا واحدا إلى اليسار

ليكن لدينا الشعاع " X "المؤلف من " n " عنصراً:

$$X = \{x_1, x_2, ..., x_n\} \qquad ; \qquad n \geq 2$$

إذا أردنا نقل عناصر هذا الشعاع موقعاً واحداً نحو اليسار مثلاً كما هو موضح في الشكل التالي:



أي أن قيمة الموقع الأول في الشعاع X هي x_2
و قيمة الموقع الثاني في الشعاع X هي x_3
وهكذا
و قيمة الموقع $n-1$ في الشعاع X هي x_n
و قيمة الموقع n في الشعاع X هي x_1

إن الخوارزمية التي تحقق النقل الدائري السابق هي:

أ - نعرف متحولا جديداً h بحيث $h = x_1$.

ب - من أجل $i = 1(1)n-1$, x_i ستأخذ القيم x_{i+1}

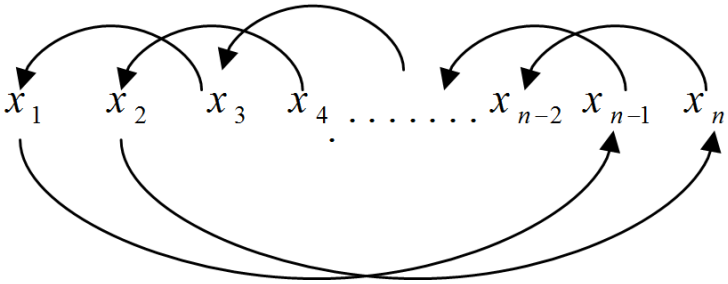
```
#include<stdio.h>
int i,h,x[10];
#define n 6
main()
{
for(i=1;i<=n;i++)
scanf("%i",&x[i]);
h=x[1];
for(i=1;i<=n-1;i++)
x[i]=x[i+1];

x[n]=h;

for(i=1;i<=n;i++)
printf("%i",x[i]);

return 0;
}
```

–النقل موقعين إلى اليسار

ليكن لدينا الشعاع " X "المؤلف من " n " عنصراً:

$$X = \{x_1, x_2, ..., x_n\} \quad ; \quad n \geq 2$$

إذا أردنا نقل عناصر هذا الشعاع موقعين نحو اليسار مثلاً كما هو موضح في الشكل التالي:



من أجل حل هذه المسألة نطبق الخوارزمية السابقة مرتين متتاليتين أي $m = 2$ فنحصل على المطلوب

```
#include<stdio.h>
int i,j,h,x[10];
#define n 8
#define m 2
main()
{
for(i=1;i<=n;i++)
scanf("%i",&x[i]);
for(j=1;j<=m;j++)
{
        h=x[1];
                for(i=1;i<=n-1;i++)
                        x[i]=x[i+1];

        x[n]=h;

        for(i=1;i<=n;i++)
                printf("%i",x[i]);
                printf("\n");
}
return 0;
}
```

ويمكننا تعميم عملية النقل السابقة أي أن ننقل عناصر الشعاع X

تطبيق 10

ترتيب مجموعة أعداد Sorting

ليكن لدينا الشعاع " X "المؤلف من " n "عنصراً:

$$X = \{x_1, x_2, ..., x_n\} \qquad ; n \geq 2$$

إن عملية ترتيب عناصر هذا الشعاع, هي مقارنة هذه العناصر بعضها فإذا كتبنا عناصر الشعاع X الأكبر فالأكبر فنكون قد رتبنا هذا الشعاع ترتيباً تنازلياً و إذا كتبنا الشعاع X الأصغر فالأصغر فنكون قد رتبنا هذا الشعاع ترتيباً تصاعدياً.

هناك عدة طرق لحل مثل هذه المسألة, سوف نوضح إحداها
إن الخوارزمية التي ترتب عناصر الشعاع X ترتيباً تنازلياً هي:

أ - نقارن العدد الأول مع كل الأعداد التالية وعددها $(n-1)$, فإذا كان هناك عدد أكبر منه, عندئذ نبدل موضعي العنصرين, ونتابع المقارنة مع بقية الأعداد مستخدمين العدد الذي أصبح في الموضع الأول حتى نصل إلى العدد ذي الترتيب " n ".

ب - العدد الثاني يقارن مع باقي الأعداد, وعددها $(n-2)$, فإذا كان هناك عدد أكبر منه, نبدل موضعي العنصرين ونتابع كما في " أ ".
وهكذا ....

ت - العدد ذو الترتيب " j " يقارن مع كل من الأعداد التي لتليه, وعددها $(n-j)$, فإذا كان هناك عدد أكبر منه, عندئذ نبدل بين موضعي العنصرين ونتابع كما في " أ ".

وتنتهي الخوارزمية, عند مقارنة العنصر ذا الترتيب $(n-1)$ مع العنصر الأخير ذا الترتيب (n).

مثال:

X		$?x_i \geq x_1$		$?x_i \geq x_2$		$?x_i \geq x_3$		$?x_i \geq x_4$			$?x_i \geq x_5$			$?x_i \geq x_6$		$?x_i \geq x_7$		$?x_i \geq x_8$	
		$j=1$		$j=2$		$j=3$		$j=4$			$j=5$			$j=6$		$j=7$		$j=8$	
x_1	5	8	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
x_2	8	5	5	7	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
x_3	3	3	3	3	3	5	7	7	7	7	7	7	7	7	7	7	7	7	7
x_4	7	7	7	5	5	3	3	4	5	6	6	6	6	6	6	6	6	6	6
x_5	2	2	2	2	2	2	2	2	2	2	3	4	5	5	5	5	5	5	5
x_6	4	4	4	4	4	4	4	3	3	3	2	2	2	3	4	4	4	4	4
x_7	9	9	8	8	7	7	5	5	4	4	4	3	3	2	2	2	3	3	3
x_8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2
x_9	6	6	6	6	6	6	6	6	6	5	5	5	4	4	3	3	2	1	1

```
#include<stdio.h>
#define n 9
int i,j,h,x[20];
main()
{
for(i=1;i<=n;i++)
        scanf("%i",&x[i]);
for(j=2;j<=n;j++)
        for(i=j;i<=n;i++)
                if(x[i]>x[j-1])
                        {
                                h=x[j-1];
                                x[j-1]=x[i];
                                x[i]=h;
                        }

for(i=1;i<=n;i++)
printf("%i",x[i]);
return 0;
}
```