المملكة العربية السعودية
جامعة الملك سعود
مركز التدريب وخدمة المجتمع

بسم الله الرحمن الرحيم

أساسيات البرمجة

إعداد الدكتور/عميد غازي

مفردات أساسيات البرمجة

1.الخوارزميات:

- مقدمة عن مفهوم الخوارزميات .
- طرق التعبير عن الخوارزمية:

- الطريقة النصية (لغة الخوارزمية Pseudo-Code)

- الطريقة البيانية (المخططات التدفقية flowchart)

- برامج وتمارين عامة.

2.لمحة عامة حول لغة ++C:

- التعرف على لغة الآلة ولغة المجمع واللغات العالية المستوى .
- التعرف على تاريخ لغة ++C.
- التعرف على البرمجة المهيكلة Structured Programming.

3.مقدمة في البرمجة بلغة ++C:

- عمليات الاسناد.
- العمليات الحسابية وعمليات المساواة والمقارنة.
- أمثلة وتمارين بسيطة.
- بنى المعطيات البسيطة (الصحيحة – الحقيقية – المحرفية – البوليانية-...)
- التصريح عن المتحولات .
- الدخل والخرج.
- برامج وتمارين عامة.

4.نبنى التحكم Control Structures:

- بنية الاختيار if.
- بنية الاختيار if/else.
- البنى التكرارية :
  - البنية While .
  - البنية do/while .
  - البنية for.
- بنية الاختيار المتعدد switch.
- برامج وتمارين عامة .

5.المصفوفات Arrays:

- التصريح عن المصفوفات .
- أمثله عن استخدام المصفوفات .
- مصفوفات الفرز.
- المصفوفات المحرفية .
- المصفوفات المتعددة الابعاد .
- برامج وتمارين عامة .

المراجع العربية:

1. كيف تبرمج بلغة ال++ c /ترجمة الدكتور صلاح الدوه جي/
2. c++ الدليل الكامل /منشورات الدار العربية للعلوم/
3. لغات البرمجة(1)/الدكتور عميد غازي/

المواقع:

www.pragsoft.com

www.cprogramming.com

www.c++Language Notes.com

www.knking.com

1-الخوارزمية algorithm

أصل كلمة خوارزمية :

إن كلمة خوارزمية مشتقة من إسم العالم العربي الجليل محمد بن موسى الخوارزمي الذي عاش في بغداد من سنة 780 الى 847م في عصر الخليفة المأمون ، وقد برع هذا العالم في الرياضيات والفلك ، وترك بصمات في التراث الحضاري العالمي ، فقد وضع الخوارزمي مبادئ علم الجبر وألف كتاب "الجبر و المقابلة" وأعطى الجبر اسمه حتى اصبحت كلمة الجبر موجودة في جميع اللغات تقريباً

وفي تلك الأونة انطلق اسم الخوارزميات Algorismsعلى جداول الضرب والقسمه والحساب العشري ، وظل هذا الاسم متداولاً في أوروبا مدة قرون حتى تطور مؤخراً ليحمل مدلولاً جديداً مرتبطاً بالبرمجة .

1. مقدمة :

إن أهم مرحلة في حل مسألة ما بإستخدام الحاسوب هي المرحلة المتعلقة بإيجاد خطة الحل ، يجب أن تكون هذه الخطة قابلة للتنفيذ من قبل الآلة ، وقابلة للتوصيف على وجهٍ لا يدعو الى اللبس أو التأويل ، يطلق اسم الخوارزمية على هذه الخطة .

2. تعرف الخوارزمية:

مجموعة الخطوات المتسلسلة والمحدودة التي تؤدي إلى حل مسألة معينة والوصول إلى نتائج محددة اعتباراً من معطيات ابتدائية.

3. أنواع الخوارزميات:

1) **خوارزميات حسابية**: تهتم بالمسائل الرياضية ٠ (حل معادلة من الدرجة الأولى)٠

2) **خوارزميات غير حسابية**: لا تهتم بالمسائل الرياضية ولكنها تحتاج إلى حل منطقي٠

(طريقة التدقيق الإملائي لنص ما،اتخاذ قراربالذهاب إلى مكان ماوتحديد الطريق الأمثل للوصول إليه)٠

سنهتم في هذا الفصل بالخوارزميات الحسابية فقط٠

4. طرق التعبير عن الخوارزمية :

- ***الطريقة الكلامية*** : كتابة الخوارزميات على شكل خطوات باستخدام اللغة المتداولة كاللغة العربية أو الإنكليزية.
- ***الطريقة الرمزية*** :كتابة الخوارزميات باستخدام الرموز.
- ***الطريقة التدفقية*** : كتابة الخوارزميات باستخدام المخططات البيانية (المخططات التدفقية).

مثال توضحى:

أكتب الخوارزمية التي تعطي نتيجة حل التعبير الرياضي الآتي باستخدام اللغة المتداولة (الطريقة الكلامية):

$$Y=(x^2+7)/x(x+2)$$

علماً بأن x معلومة ٠

الحل:

يمكن التعبير عن الخوارزمية باللغة المتداولة(العربية) على الشكل الآتي:

الخطوة الأولى :أقرأ(أدخل) قيمة المتحول x .

الخطوة الثانية: احسب المقام : $a=x(x+2)$

الخطوة الثالثة: إذا كان المقام مساوياً للصفر اطبع " المسألة ليس لها حل" ٠

الخطوة الرابعة: وإلا احسب البسط : $b=(x^2+7)$

الخطوة الخامسة:احسب قيمة y : $y = a / b$ ٠

الخطوة السادسة: اطبع (أكتب) قيمة y٠

الخطوة السابعة: توقف ٠

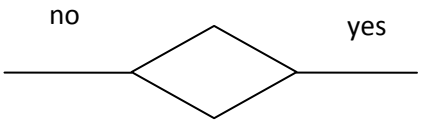
المخطط التدفقى (الهندسى أو الكابى):

لبناء المخطط التدفقي نستخدم مجموعة من الأشكال الهندسية لتسهيل هذا المخطط :

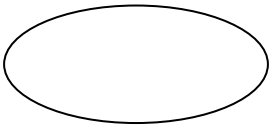
1. عمليات الإدخال والإخراج نستخدم الشكل :

2. عملية المعالجة نستخدم الشكل :

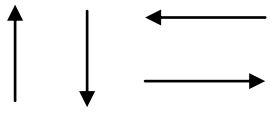
3. عملية الشرط(القرار) نستخدم الشكل :



4. لبداية ونهاية الخوارزمية نستخدم الشكل :



5. لمعرفة اتجاه الخوارزمية نستخدم الشكل :



6. نقطة توصيل وربط :

تمرين1: اكتب الخوارزمية الكلامية والرمزية والمخطط التدفقي لإيجاد مساحة ومحيط المستطيل ؟.

الحل :

الخوارزمية الكلامية :

1- المدخلات : الطول والعرض .

2 - المعالجة : المساحة(s) = الطول x العرض

المحيط (m) (الطول+ العرض) x 2

3- المخرجات : المساحة والمحيط

الخوارزمية الرمزية :

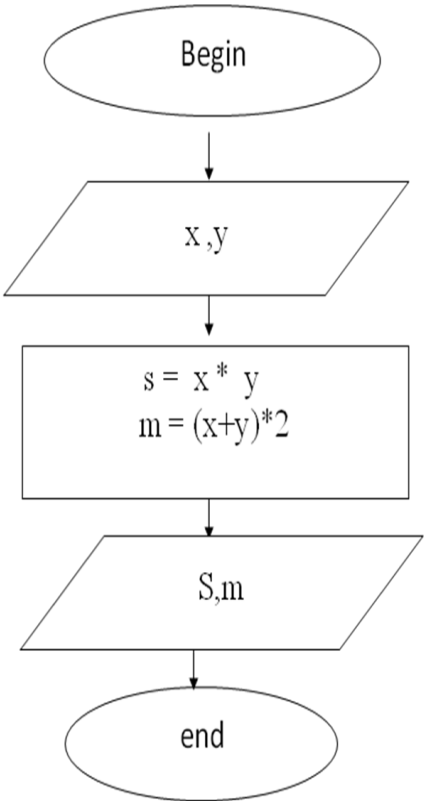
1- المدخلات : y, x

2- المعالجة : s= y*x

m = (y+x) * 2

3- المخرجات : m , s

المخطط التدفقي :



***تمرين2*: على نمط المثال السابق اكتب الخوارزمية الكلامية و الرمزية والمخطط التدفقي لإيجاد مساحة ومحيط الدائرة ؟**

الحل :

الخوارزمية الكلامية :

1. المدخلات: نصف القطر (r)
2. المعالجة : المساحة(s) = π x نصف القطر للتربيع

   المحيط (m) =2 x نصف القطر p x
3. المخرجات: المساحة والمحيط لدائرة

الخوارزمية الرمزية :

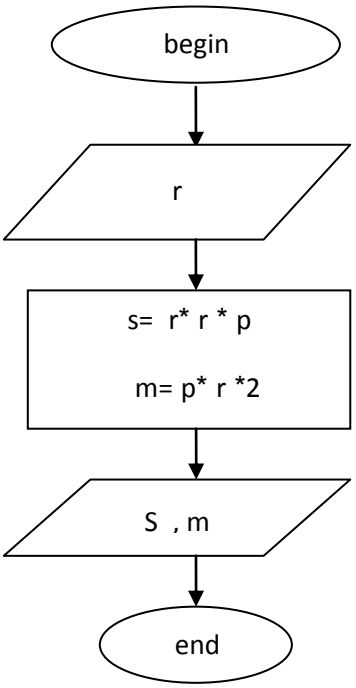
1- المدخلات : r

2- المعالجة : s= p* r * r

m= p* r *2

3- المخرجات : s,m

المخطط التدفقي :



تمرين3: اكتب الخوارزمية الرمزية والمخطط التدفقي لإدخال x (عدد) وإيجاد قيمة y=(x-2)/x

الحل:

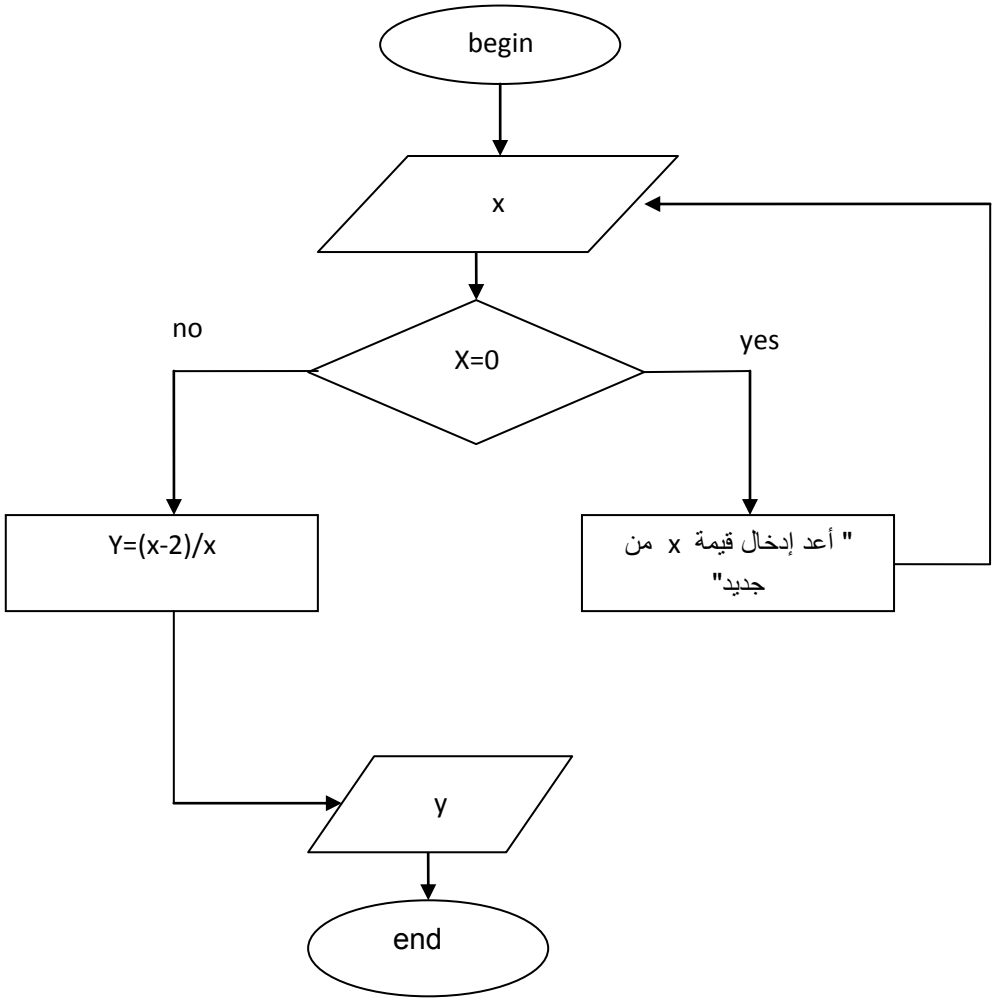
الخوارزمية الرمزية : 1) المدخلات : x

2) المعالجة : إذا كانت x=0 عندئذ " أعد ادخال قيمة x من جديد لانه لايمكن القسمة على صفر"

والا فاحسب : y= (x-2)/x

3) المخرجات : y

المخطط التدفقي :



تمرين4: اكتب الخوارزمية الرمزية والمخطط التدفقي لايجاد y=x/(x-3)

الحل:

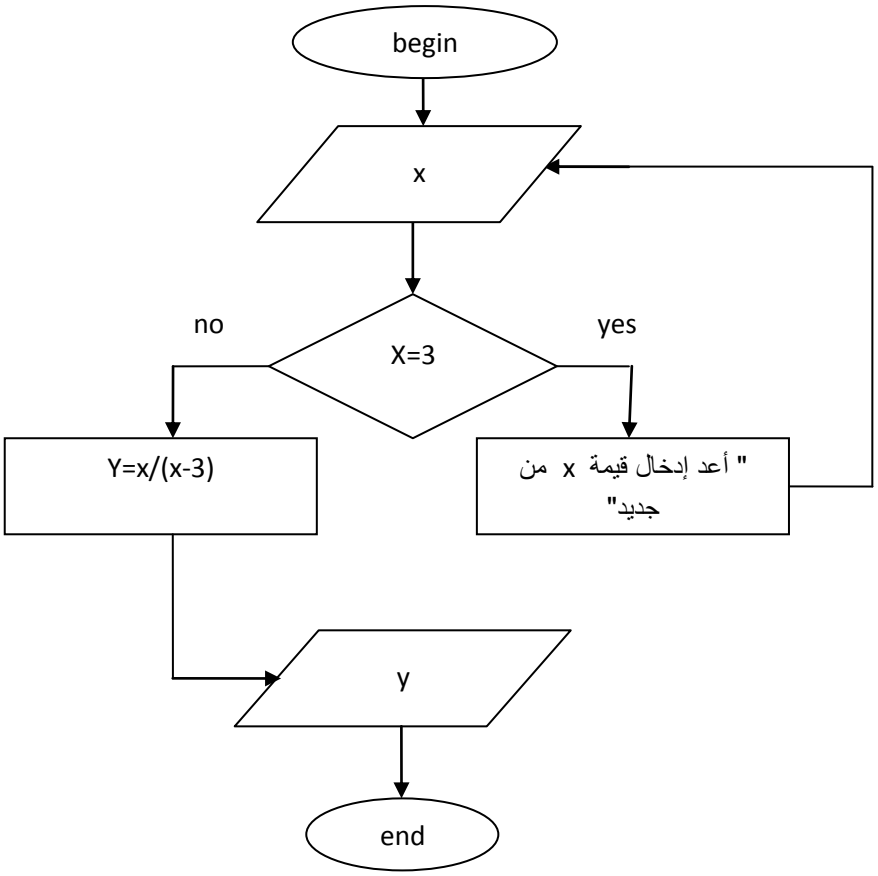
الخوارزمية الرمزية : 1) المدخلات : x

2) المعالجة : إذا كانت x=3 عند ئذ " اعد ادخال قيمة x "

والا اطبع y=x/(x-3)

3) المخرجات : y

المخطط التدفقي :



تمرين 5: اكتب الخوارزمية الرمزية والمخطط التدفقي لحل المعادلة :a X + b = 0

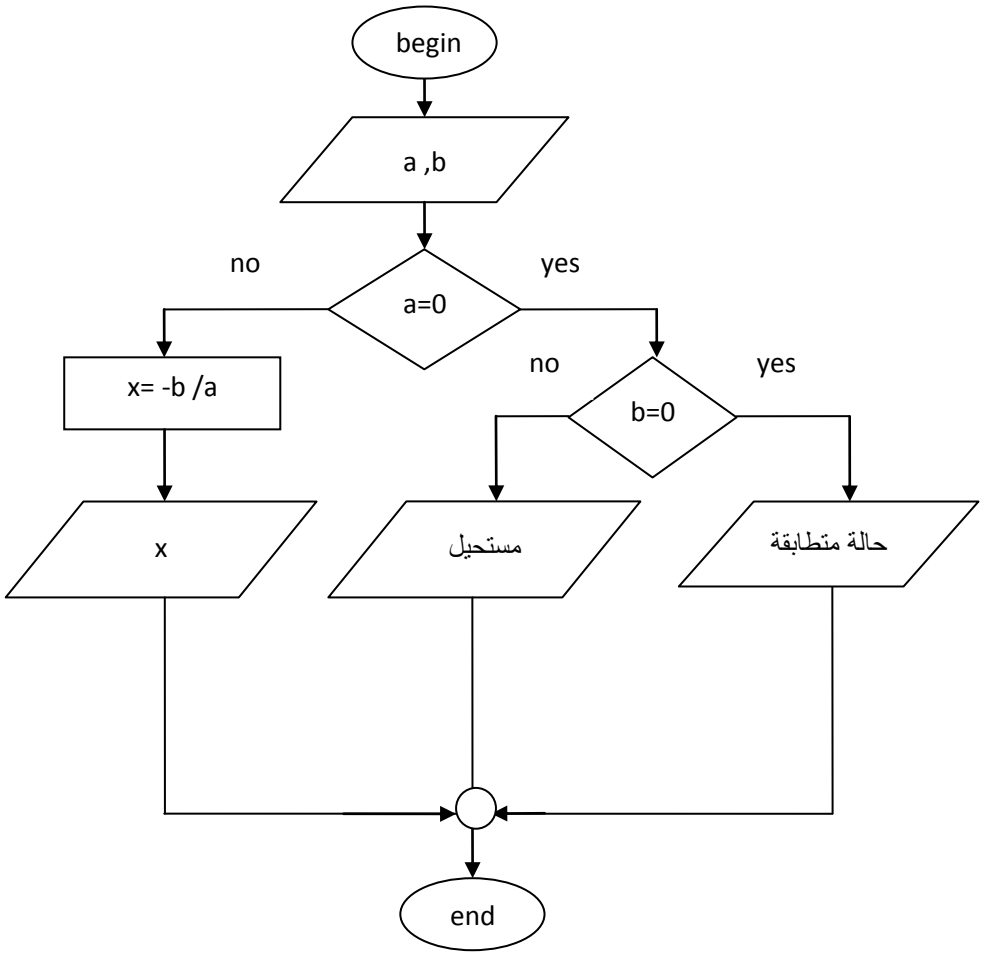
مناقشا جميع الحالات الممكنة لـ a,b

الحل :

الخوارزمية الرمزية :

- أدخل(اقرأ): a ، b .
- إذا كان (a=0 , b< >0) نجد : 0x+b=0 أي b=0
أطبع (أكتب) : "مستحيل الحل"٠
- إذا كان (a=0 , b=0) نجد : 0x+0=0
أطبع (أكتب) : " حالة متطابقة "
- إذا كان (a< >0) نجد : x=-b/a
أطبع قيمة x

المخطط التدفقي :



تمرين 6: اكتب الخوارزمية الرمزية والمخطط التدفقي (الانسيابي) لإيجاد قيمة y المعطاة بالشكل التالي :

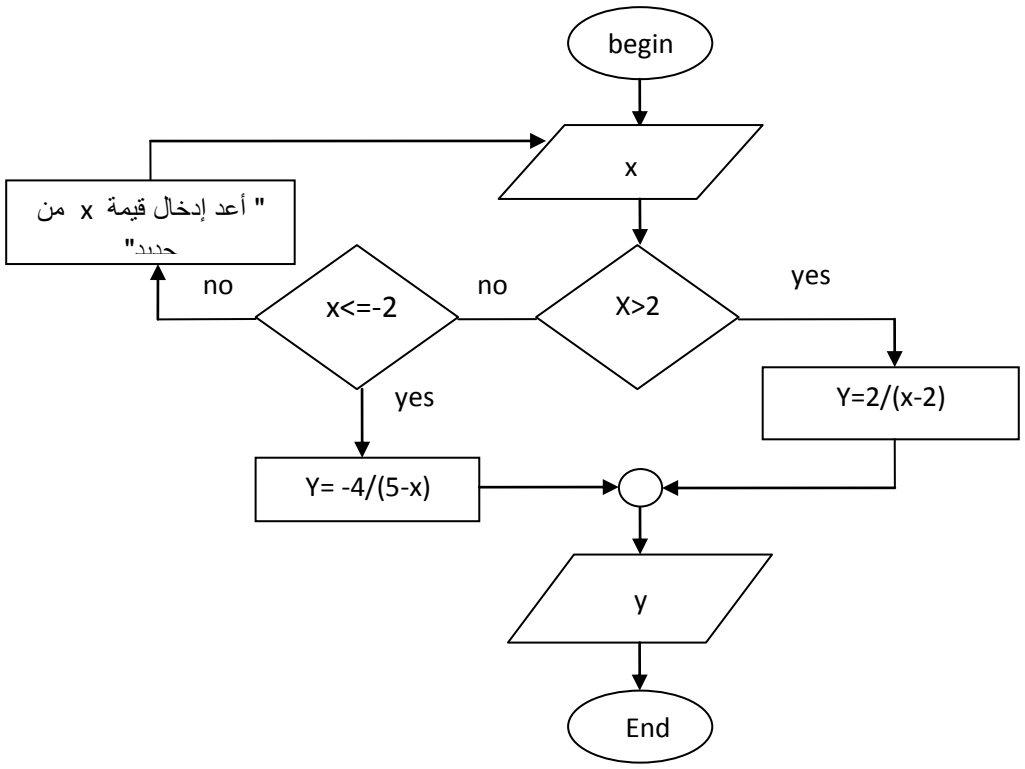
$$Y=\begin{cases} 2/(x-2) & x>2 \\ -4/(5-x) & x<=-2 \end{cases}$$

الحل :

الخوارزمية الرمزية :

1- المدخلات : x
2- المعالجة : إذا كانت x>2 عندئذ y=2/(x-2)
وإلا إذا كانت x<=-2 عندئذ y=-4/ (5-x)
وإلا " اعد ادخال قيمة x "
3- المخرجات : y

المخطط الانسيابي (التدفقي ، الصندوقي) :



تمرين 7: اكتب الخوارزمية والرمزية والمخطط التدفقي لحل معادلة الدرجة الثانية

$$aX^2+bX+c=0$$

الحل :

الخوارزمية الرمزية :

1) أدخل (اقرأ) : a,b,c

2) إذا كان (a=0) نفذ :

تصبح المعادلة معادلة من الدرجة الأولى : bX+c=0

i) إذا كان (b = 0) نفذ :

C =0 حالة متطابقة

C < >0 حالة مستحيلة

ii) إذا كان (b <> 0) نفذ :

X= -c/b

٣) إذا كان (a<> 0) نفذ :

حساب D دالتا : $D=b^2 -4*a*c$

i) إذا كان (D = 0) نفذ :

أطبع : " للمعادلة جذران متماثلان "

وأحسب: $X_1 = X_2=-b/2*a$

ii) إذا كان (D < 0) نفذ :

أطبع : " للمعادلة جذران عقديان "

iii) إذا كان (D > 0) نفذ :

أطبع : " للمعادلة جذران حقيقيان "

وأحسب: $X_1=(-b-sqrt(D))/(2*a)$

$X_2=(-b+sqrt(D))/(2*a)$

المخطط التدفقي:

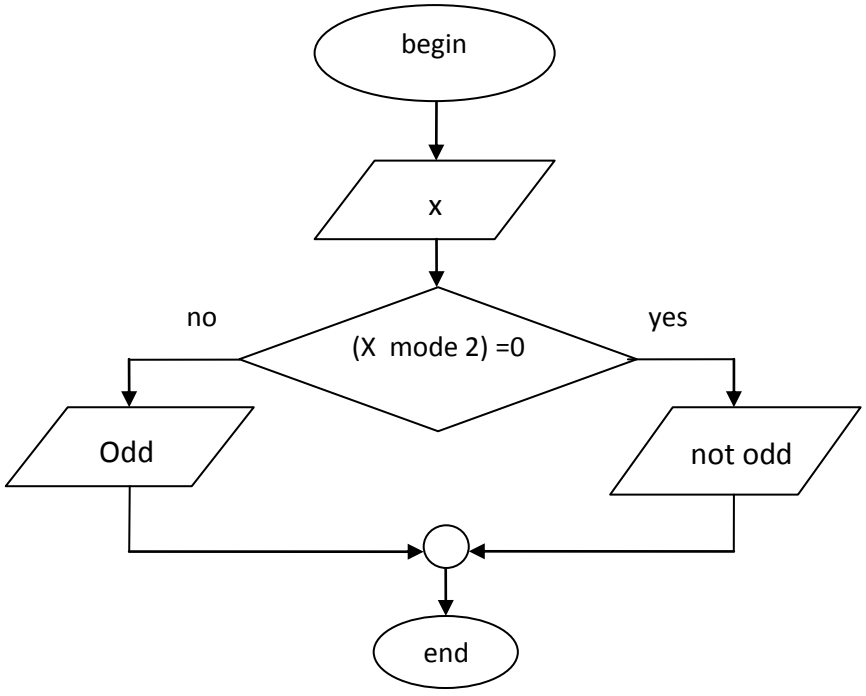
تمرين 8: اكتب الخوارزمية الرمزية والمخطط التدفقي لإدخال عدد صحيح (x) موجب وطباعة إذا كان فرديا أم زوجيا ؟

الحل :

الخوارزمية الرمزية :

1. المدخلات: x
٢. المعالجة والمخرجات: إذا كان باقي قسمة العدد على 2 يساوي صفر (x mode 2= 0) فإن :
أطبع " العدد زوجياً أو not odd "
وإلا فإن :
أطبع " العدد فردياً أو odd "

المخطط التدفقي :



تمرين 9: اكتب الخوارزمية الرمزية والمخطط التدفقي لإدخال عشرة أعداد مختلفة وإيجاد المتوسط والمجموع ؟

الحل :

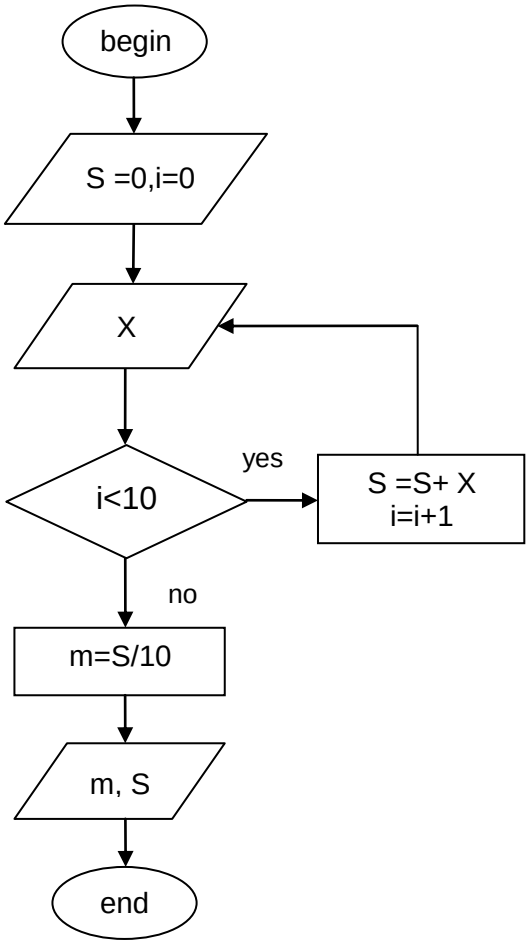
الخوارزمية الرمزية :

1. المدخلات : x ؛ S =0 ؛ i=0
2. المعالجة : العداد (i=i+1); المجموع (S=S+x)
إذا كان i<10 عندئذ " أعد إدخال "
وإلا i >=10 عندئذ " توقف عن الإدخال i "

m=S/10

3. المخرجات : المجموع (s) ، المتوسط (m)

المخطط التدفقي :



تمرين10: اكتب الخوارزمية الرمزية والمخطط التدفقي لإدخال عشرة أعداد وطباعة الفردي منها فقط ؟

الحل :

الخوارزمية الرمزية :

1- المدخلات : X, i=0

٢- المعالجة و المخرجات: إذا كان i<10 عندئذ

وإذا كان X mode 2=0 عندئذ

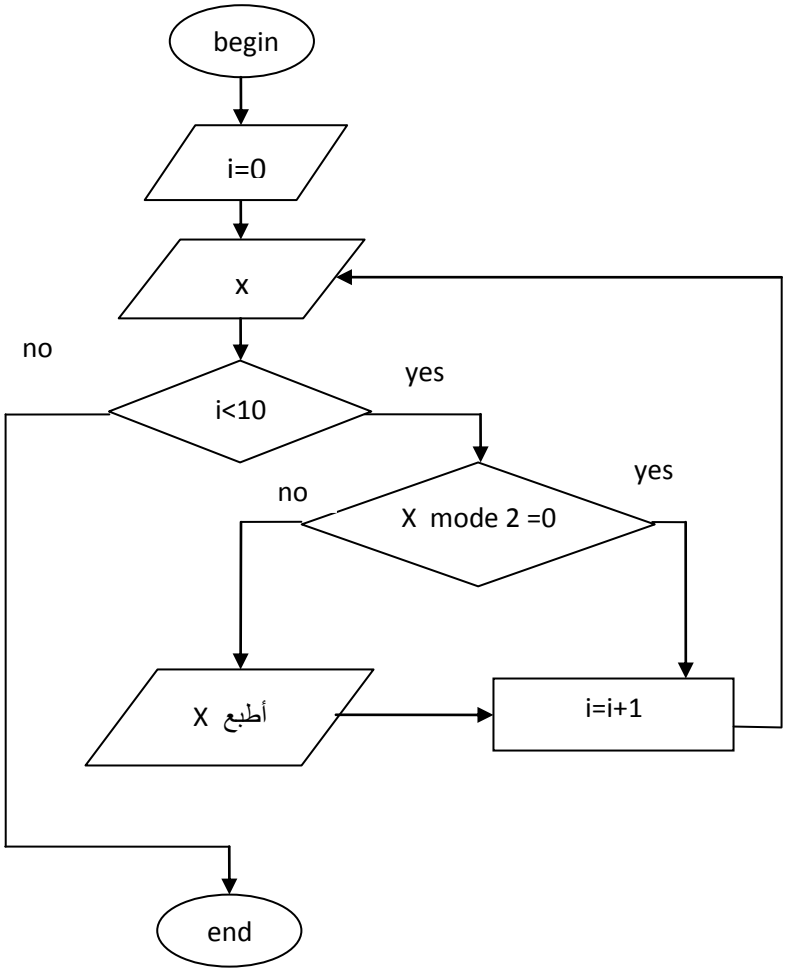
i=i+1 و" أعد إدخال x "

وإلا أطبع() أكتب قيمة x الحالية ثم أدخل قيمة جديدة لـ x

i=i+1 وشغل العداد.

وإلا أخرج من البرنامج

المخطط التدفقي :



مقدمة في البرمجة بلغة C++

* أنواع اللغات :

يمكن تقسيم اللغات المستخدمة في البرمجة إلى ثلاثة أنواع:

1- لغة الآلة ، 2. لغة المجمع 3. اللغات العالية المستوى

1. لغة الآلة:

هي اللغة التي يستطيع الحاسب أن يفهمها مباشرة وهي معرفة من قبل البنية الصلبة للحاسب ، تتألف بشكل عام من سلاسل من الأعداد (مجموعات من الأصفار والواحدات) التي تعطي الأوامر للحاسب من أجل تنفيذ تعليماته الأولية كل تعليمة على حده.

ترتبط هذه اللغة ارتباطاً وثيقاً بالآلة machine-dependent وهذا يعني أن لغة آلة ما لا تستخدم إلا لنفس النوع من الآلات فقط .

2. لغة المجتمع :

هي لغة تستخدم مصطلحات قريبة من اللغة الإنكليزية للتعبير عن العمليات الأولية للحاسب، وقد تم تطوير مترجمات للبرامج تسمى بالمجمعات assemblers تحويل البرامج من لغة المجمع إلى لغة الآلة.

3. اللغات العالية المستوى :

هي اللغات التي ظهرت لتسريع عملية البرمجة وذلك باستخدام تعليمات تقوم بالعديد من المهام الجوهرية ، وتهد اللغات C++,C من أكثر اللغات العالية المستوى قوة وانتشاراً .

- تدعى البرامج التي تقوم بتحويل النصوص من البرامج مكتوبة بلغات عالية المستوى إلى لغة الآلية بالمترجمات.

ملاحظات:

1- ÷ يوجد بعض المفسرات interpreter programs التي تستطيع تنفيذ البرامج المكتوبة بلغات عالية المستوى مباشرة دون الحاجة إلى ترجمة هذه البرامج إلى لغة الآلة.

2- البرامج المترجمة هي أسرع تنفيذاً من البرامج المفسرة عموماً .

* البرمجة بلغة C++:

تسهل لغة C++ الأسلوب المهيكل والمنهجي لعملية تصميم البرامج ، حيث تتألف برامج هذه اللغة من مكونات تسمى الصفوف classes والتوابع Functions وبالتالي يمكن تقسيم عملية تعلم لغة الـC++ إلى قسمين : يعتمد الأول منها على تعلم لغة الـC++ نفسها في حين يسمح الثاني بتعليم كيفية استخدام الصفوف الملحقة بهذه اللغة واستخدام التوابع الموجودة ضمن المكتبة المعيارية ANSI C.

*** مراحل تنفيذ برامج C++ :**

يتم التنفيذ خلال ست مراحل هي بالشكل التالي:

- **مرحلة الكتابة ضمن Edit** : وهي كتابة نص البرامج في أي محرر نصوص يستخدم لكتابة البرامج بلغة C++ .
- مرحلة ما قبل الترجمة ؛ Preprocess : هي تصحيح البرنامج من الأخطاء ومن ثم تخزينه على وحدة تخزين ثانوية مثل الأقراص بتوسع CPP, CXX وذلك حسب بيئة العمل.
- **مرحلة الترجمة Compile** : هي ترجمة البرنامج إلى لغة الآلة.
- **مرحلة الوصل Linking**: تتضمن برامج الـ C++ استدعاءات لتوابع تم تعريفها في مكان آخر مثل المكتبات المعيارية ، وبالتالي مهمة هذه المرحلة هي استخدام الواصل Linker لوصل الملف مع نصوص التوابع الناقصة من أجل الوصول إلى صورة قابلة للتنفيذ .
- **مرحلة الشحن Loading** : قبل تنفيذ البرنامج يجب وضعه في الذاكرة وذلك باستخدام الشاحن Loader الذي يقوم بأخذ الملف التنفيذي ونقله إلى الذاكرة.
- **مرحلة التنفيذ Execute**: هي مرحلة التنفيذ التي تتم تحت إشراف وسيطرة وحدة التحكم والمعالجة CPU .

* أمثلة بسيطة : لتعلم مبادئ أساسية في لغة ++ C –

1– طباعة نص مؤلف من سطر :

كل الكتابات التي تلي هذه الإشارة (//) تسمى تعليق لا يتم تنفيذه

```
// First Program
# include<iostream.h>
```

// (.h) توجيه ما قبل الترجمة حيث يتم ضمن محتوى

الملف الرأسي ذو الامتداد الحاوي على العمليات الخاصة بالدخل والخرج لنص البرنامج

```
main ( )
{
cout <<  " welcome to c++ " ;
return 0 ;
}
```

// التابع الرئيسي الذي يبدأ من عند التنفيذ

// بداية البرنامج

// تعليمة الطباعة

// إحدى طرق الخروج من التابع

// نهاية البرنامج

```
(Inactive C:\TCWIN45\BIN\NONAME00.EXE)
welcome to c++
```

2 . برنامج جمع عددين صحيحين :

```
# include < iostream.h>
main ( )
{
  int x1 , x2, x3 ;
  cout <<" enter first numbe ";
  cin >> x1 ;
```

// تعريف المتحولات

// تعليمة الطباعة

// تعليمة قراءة متحول

```
  cout << " enter second number ";
  cin >> x2
  x3 = x1 + x2 ;                    //x3 إجراء عملية الجمع والإسناد إلى المتحول الجديد
  cout << "sum is " <<x3 ;          // تعليمة الطباعة المتعددة
return 0 ;
}
```

```
(Inactive C:\TCWIN45\BIN\NONAME01.EXE)
enter first numbe 10
enter second number 55
sum is 65
```

ملاحظة:

1- مفهوم يتعلق بالذاكرة ألا وهو طريقة حجز المتحولات:

كل اسم من أسماء المتحولات مثل .... , x3 , x2 ,x1 يتم وضعه في الذاكرة ويعرف بإسم name ونمط type وحجم size وقيمة value وبالتالي فإن المتحول x1 يملك الاسم x1 يملك الاسم x1 والنمط int والحجم 2 بايت والقيمة هي حسب القيمة المقروءة .

x1	5
x2	10
x3	15

مواضع المتحولات في الذاكرة مع ذكر الاسم والقيمة

2. أنواع المتحولات:

- المتحول التعدادي enum
- المتحول المحرفي char
- المتحولات الصحيحة short int, int , long int , unsigned sort int, unsigned int , unsigned long int.
- المتحولات الحقيقة float , double , long double

وبين الجدول التالي أنواع المتحولات ومجالاتها :

نوع المتحول	المجال
char	−128 to 127
int	−32768 to 32767
unsigned int	0 to 65535
short int	−32768 to 32767
Unsigned short int	0 to 65535
Long int	−2147483648 to 214748364
float	−3.4E−38 to 3.45E+38
double	−1.7E−308 to 1.7E+308
long double	−3.4E−4932 to 1.1E+4932

* **العمليات الحسابية** :

العملية	اسم العملية	ترتيب عملية التقسيم (الأولوية)

اسم العملية	الرمز الحسابي	طريقة التعبير حسب لغة C++
الجمع	+	x1 + x2
الطرح	−	x2 − x1
الضرب	*	x1 * x2
القسمة	/	x1 / x2
باقي القسمة الصحيحة	%	x1 % x2

تقوم C++ بتطبيق العمليات في العبارات الحسابية حسب ترتيب معين محدد تبعاً لقواعد الأولوية بين العمليات التي تماثل قواعد الأولوية في الجبر وذلك كما في الجدول التالي:

( )	الأقواس	**تقييم أولاً ، إذا وجد في العبارات الحسابية أقواس متداخلة ضمن بعضها البعض فالحساب يبدأ انطلاقاً من أول مجموعة في الداخل أما إذا كان لدينا مجموعة من الأقواس جانب بعضها البعض وعلى نفس المستوى عندها يبدأ الحساب من اليسار إلى اليمين.**
* ، / ، %	الضرب، القسمة ، باقي القسمة	**تقييم ثانياً ، إذا وجدت على نفس المستوى فإنها تقييم من اليسار إلى اليمين .**
+ ، −	**الجمع ، الطرح**	**تقييم في النهاية ، إذا وجدت على نفس المستوى فإنها تقييم من اليسار إلى اليمين .**

أما بالنسبة لعمليتي الإسناد والمقارنة فتتم بالشكل التالي: جميع العمليات الحسابية يتم تجميعها من اليسار إلى اليمين إلا عملية الإسناد تتم من اليمين إلى اليسار .

الشكل الجبري	الشكل الموافق حسب c++	مثال	معنى الكتابة
=	= =	x = = y	x تساوي y
≠	! =	x != y	x لا تساوي y
<	<	x <y	x أصغر من y
>	>	x >y	x أكبر من y
≤	<=	x < = y	أصغر أو يساوي y
≥	>=	x > = y	x أكبر أو يساوي y

***العملية المنطقية Logical operators :**

وهي ثلاثة :

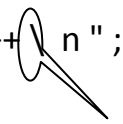
And يرمز لها &&

Or يرمز لها ||

Not يرمز لها !

*** سلاسل الهروب :**

```
# include <iostream.h>
main ( )
{
  Cout <<"welcome to c++\ n " ;
```



حرف الهروب

```
return 0;
}
```

يدعى \ بحرف الهروب وهو يلحق بحرف يدل على معنى معين كما هو موضح في الجدول:

سلسلة الهروب	المعنى
\ n	سطر جديد أي وضع المؤشر في بداية السطر التالي
\ t	تحريك المؤشر مسافة جدولية أفقية
\"	تستخدم لطباعة علامة الاقتباس

*** بعض الأمثلة:**

1- أكتب برنامجاً يأخذ كدخل ثلاث أعداد صحيحة من لوحة المفاتيح ثم يطبع مجموعها ومتوسطها وناتج جداؤها.

```
# include < iostream.h>
main ( )
{
int a , b, c ;
cout << " enter a =" ; cin >> a ;
cout << " enter b = " ; cin >> b ;
cout << " enter c = " ; cin >> c ;
cout << " sun is " << a+b+c << " \ n" ;
cout << average is " << ( a+b+c)/3 <<" \n";
cout << product is " << a * b* c;
return ;
}
```

```
(Inactive C:\TCWIN45\BIN\NONAME02.EXE)
enter   a = 10
enter   b = 20
enter   c = 33
sun is          63
average is      21
product is      6600
```

2- أكتب برنامج يقرأ نصف قطر دائرة ثم يطبع قيمة قطر الدائرة ، محيطها ، مساحتها .

ملاحظة : قيمة π =3.14

```
# include <iostream.h>
main  ( )
{
float  r ;                    // تعريف متحول حقيقي
float p = 3 , 14 ;            // تعريف متحول حقيقي وإسناد قيمة له
cout << " enter r =" ; cin >> r ;
cout << r * 2=" << r * 2<<"\n";
cout <<"2*p*r = " << 2*p*r<<"\n" ;
cout << "p*r*r =" << p*r*r;
return 0 ;
}
```

```
(Inactive C:\TCWIN45\BIN\NONAME02.EXE)
enter  r = 4.5
r * 2 = 9
2 * p * r = 28.26
p*r*r = 63.585
```

3- أكتب برنامجاً يقوم بطباعة مستطيل

```
# include < iostream.h>
main ( )
{
```

```
  cout << " ***********\n" ;
  cout << " *\t " <<" *\n";
  cout << " *\t " <<" *\n";
  cout << " *\t " <<" *\n";
  cout << "***********\n";
return 0;
}
```

```
  (Inactive C:\TCWIN45\BIN\NONAME04.EXE)
**********
*          *
*          *
*          *
**********
```

3.بنى التحكم Control Structures:

1.البنى الشرطية:

* بنية الأختيار if :

تقوم بنية الاختيار ifبتنفيذ فعل معين عندما يكون الشرط المرافق لها محققاً وإلا يتم تجاهله ، ولها الشكل العام التالي :

if (condition) statement :

مثال 1:

علامة النجاح في أحد الامتحانات تساوي 60 درجة عندها فإن تعليمة الـ if تكون بالشكل:

```
if  ( grad > = 60 ) cout <<"passed";
```

مثال 2:

أكتب برنامجاً يطلب من المستخدم إدخال عددين صحيحين .ثم يأخذ العددين ليطبع العدد الأكبر بينهما متبوعاً بالرسالة "is larger" إذا كان العددان متساويين عندها يطبع البرنامج الرسالة "the number are equal"

```
#include <iostream.h>
main ()
{
    Int a, b;
    cout<<"enter" a=";cin>>a;
    cout<<"enter" b=";cin>>b;
    if ( a > b ) cout <<a<< " is larger" ;
```

```
        if ( a < b ) cout <<b<<" is larger" ;
        if ( a = = b ) cout <<"the numbers are eonal"
        return 0 ;
    }
```

```
Inactive C:TCWIN45\BIN\NONAME05.EXE)
enter  a = 100
enter b = 69
100 is larger .
```

* بنية الأختيار if/else

تسمح بنية الاختيار if / else بتحديد جملة من الأفعال الممكن تنفيذها إذا كان الشرط المرافق صحيحاً أو إذا لم يكن كذلك ، ولها الشكل العام التالي:

```
if ( condition )
    statement 1 ;
else
    statement 2;
```

مثال 1:

إذا كان علامة الطالب أكبر أو يساوي القيمة 60 درجة فيطبع كلمة "passed" وإلا فهي تطبع الكلمة "failed" عندها فإن تعليمة الـ if / else تكون بالشكل

```
    if ( grad > = 60 )
        cout << " passed " ;
    else
        cout << "failed" ;
```

مثال 2:

أكتب برنامجاً يقرأ عدداً صحيحاً ثم يحدد و يطبع فيما إذا كان هذا العدد زوجياً أم فردياً .

```
# include < iostream.h>
main ()
{
    int a ;
    cout <<"enter a ="; cin>>a;
    if ( a % 2 = = 0)
        cout << " not odd" ;
    else
        cout << " odd" ;
    return 0 ;
}
        Inactive C:\TCWIN45\BIN \ NONAME06.EXE)
        enter a = 13
        odd
```

ويمكن استخدام البني if / else المتداخلة من أجل القيام بفحص عدة حالات من خلال وضع البني

if / else داخل بعضها البعض . على سبيل المثال إذا كانت علامة الفحص أكبر أو يساوي 90 فيتم طباعة الحرف a وإذا كانت بين 89 و 80 فتطبع الحرف b وإلا فيتم طباعة الحرف c . وبالتالي تكون العملية C++ المكافئة بالشكل:

```
if ( grad > = 90 )
        cout << "a" ;
else if ( grad >= 80)
        cout << "b" ;
else
```

```
cout << "c" ;
```

ملاحظة :

عادة تضع تعليمة واحدة في جسم البنية الاختيارية if ولكن إذا أردنا وضع عدة تعليمات يجب أن نقوم بوضعها داخل قوسين كبيرين ({ }) . نسمى مجموعة التعليمات المحتواه ضمن زوج من الأقواس الكبيرة بالتعليمية المركبة compound statement .

مثال 1:

```
if (grad > = 60 )
        cout << " passed" ;
else
        {
        cout << " failed " ;
              cout << " you must take this course again" ;
        }
```

في هذه الحالة إذا كانت قيمة gradأصغر من 60 عندها يقوم البرنامج بتفيذ التعليمتين الموجودتين في الجزء else ويطبع ما يلي:

```
failed
you must take this course again
```

<u>بعض الأمثلة:</u>

1- أكتب برنامج يأخذ كدخل عددين صحيحين من لوحة المفاتيح ويفحص فيما إذا كان الثاني قاسم للأول.

```
# include<iostream.h>
main ( )
{
        int  a , b ;
        cout<<"enter a=";cin>>a;
```

```
        cout<<"enter b=;cin>>b;
        if ( b! = 0 && a % b = = 0 )
                cout << a << ' is divisible by " <<b ;
        else
                cout <<a<<is not divisible by " << b ;
        return 0 ;
}
```

```
        Inactive C\TCWIN45\BIN\NONAME00.EXE)
        enter a = 25
        enter b = 5
        25 is divisible by 5
```

2- أكتب برنامج يأخذ كدخل ثلاث أعداد صحيحة ثم يطبع أصغر هذه الأعداد.

```
# include < iostream.h>
main ()
{
  int a , b, c ;
  cin >> a >> b >> c ;
  if ( a > b )
        if (a < c ) cout << " min is" << a ;
        else cout << " min is " << c ;
  else
        if ( b < c ) cout << "min is " << b ;
  else
```

```
        cout << " min is " << c ;
   return 0;
}
        Inactive C:\TCWIN45\BIN\NONAME01.EXE)
        enter  a = 10
        enter b = 8
        enter c = 77
        min is 8
```

---

2. البنى التكرارية :

- **البنية التكرارية While** :

تسمح البنية التكرارية للمبرمج بتحديد مجموعة من الأفعال يجري تكرارها طالما ظل الشرط المرافق للبنية محققاً ، ولها الشكل العال التالي :

```
while (condition )
statement
```

مثال 1:

أكتب برنامج لطباعة الأعداد من 1−10 بشكل عمود واحد.

```
 # include < iostream.h>
 main ()
 {
        int i ;
        i = 1 ;
        while ( i <=10)
```

```
        {
                cout << i << "\n" ;
                i = i +1 ;
        }
        return 0 ;
}
```

```
(Intactive C:\TCWIN45\BIN\NONAME02.EXE)
1
2
3
4
5
6
7
8
9
10
```

مثال 2:

يفرض لدينا علامات مذاكرة قام بها طلاب صف مؤلف من عشرة طلاب والمطلوب حساب معدل علامات طلاب الصف في هذه المذاكرة .

```
# include < iostream.h>
main ()
{
        float mark , sum ;
```

```
        int  i = 1
        sum = 0
        while ( i < = 10 )
        {
                cout<<"enter the mark=";cin>>mark;
                sum = sum + mark ;
                i = i +1 ;
        }
        cout<<"average is : "<<sum/10 ;
        return 0 ;
}
```

```
(Intactive C:\TCWIN45\BIN\NONAME03EXE)
enter   the     mark    = 13
enter   the     mark    = 44
enter   the     mark    = 54
enter   the     mark    = 60
enter   the     mark    = 90
enter   the     mark    = 33
enter   the     mark    = 75
enter   the     mark    = 56
enter   the     mark    = 55
enter   the     mark    = 78
        average is : 55.8
```

ملاحظة:

1- إن عدم وضع تعليمة أو فعل جسم البنية while يسبب عدم تحقق الشرط المرافق لها وينتج عن ذلك عدم إنتهاء التكرار .

2- تسبب كتابة الكلمة while مع حرف كبير في البداية خطأ وذلك على اعتبار أن لغة C++ حساسة لحالة الحروف تحتوي كافة الكلمات المفتاحية الخاصة بلغة C++ مثل if , while ، .. وغيرها على شكل حروف صغيرة .

3- إن أي متحول لا يعطي قيمة ابتدائية يمكن أن يكون له قيمة ما لا يعرف عنها شيء مخزنة مسبقاً في موضع الذاكرة المخصص لهذا المتحول ، وبالتالي إن عدم إعطاء متحول حساب مجموع مثل sum أو عداد مثل I سوف يؤدي إلى الحصول على نتائج قد تكون خاطئة.

• البنية التكرارية do / while :

تشبه بنية التكرار do / while البنية while حيث نقوم بالبنية while بالتحقق من صحة شرط الاستمرار بالتكرار في بداية الحلقة قبل تنفيذها ، أما في حالة البنية do / while فيتم ذلك بعد تنفيذ جسم الحلقة أولاً . أي يتم تنفيذ جسم البنية do / while مرة واحدة على الأقل. عند الإنتهاء من تنفيذ البنية do / while يتم الانتقال إلى التعليمية التي تلى مباشرة جزأها while ، ولها الكل العام التالي:

```
do
  statement ;
while ( condition ) ;
```

وقد تم استخدام الأقواس الكبيرة لتحديد جسم البنية do / while حتى لا يتم الخلط بين البنيتين while , do / while ، لذلك يتم عادة كتابة البنية do / while على الشكل التالي:

```
do {
  statement
} while ( condition )
```

مثال 1:

نفس المثال السابق . أكتب برنامج لطباعة الأعداد من 1-10 بشكل عمود واحد . ولكن باستخدام

do / while

```
# include <iostream.h>
main()
{
        int i ;
        i = 1
        do {
                cout <<i<<"\n";
                i=i+1;
            } while ( i <=10);
        return 0;
}
```

```
(Intactive C:\TCWIN45\BIN\NONAME02.EXE)
1
2
3
4
5
6
7
8
9
10
```

* بنية التكرار for:

تدعى هذه البنية أيضاً بالبني التكرارية ذات العداد ، وهي تتطلب ما يلي:

1- تعريف متحول التحكم بالحلقة (وهو عداد الحلقة)

2- تحديد القيمة الابتدائية لمتحول التحكم بالحلقة .

3- تحديد أسلوب الزيادة (أو الانقاص) الذي يتم من خلاله تغيير قيمة متحول التحكم بالحلقة في كل مرة نمر فيها.

4- تحديد الشرط الذي من خلاله نقوم بفحص النتيجة النهائية لمتحول التحكم بالحلقة (حتى نحدد إذا كان من الممكن معاودة تنفيذ الحلقة).

ولها الشكل العام التالي:

```
for ( exp 1;     exp 2  ; exp 3 )
statement ;
```

exp 1 : يمثل تعريف وتحديد القيمة الابتدائية لعداد الحلقة.

exp 2 : يمثل شرط إنهاء الحلقة أي شرط فحص النتيجة النهائية لعداد الحلقة.

exp 3 : يمثل أسلوب زيادة أو إنقاص عداد الحلقة.

مثال1:

نفس المثال السابق . أكتب برنامج لطباعة الأعداد من 1-10 بشكل عمود واحد . ولكن بإستخدام البنية for .

```
# include <iostream.h>
main()
{
  for ( int I= 1; I<=10 ; I=I+1)
        cout <<I<<"\n";
  return 0;
}
```

```
(Intactive C:\TCWIN45\BIN\NONAME02.EXE)
1
2
3
4
5
6
7
8
9
10
```

مثال 2:

أكتب برنامج لحساب مجموع جميع الأعداد الصحيحة من 2 إلى 100

```
# include < iostream.h>
man ( )
{
        inst sum = 0 ;
        for ( int i = 2 ; i < = 100 ; i = i +1)
                sum  = sum + i ;
        cout << " sum is " << sum ;
        return 0 ;
}
```

```
(Intactive C:\TCWIN45\BIN\NONAME02.EXE)
Sum is 5049
```

عمليات الإسناد:

يتوفر في لغة c++ عدداً في من عمليات الإسناد المختصرة التي هي تعبير على عملية الإسناد نفسها، فعلى سبيل المثال يمكن اختصار التعليمية التالية:

```
c=c+3;
```

لتصبح بالشكل التالي:

```
c+=3;
```

حيث نسمي العملية =+ بعملية الإسناد والجمع addition assignment operator

وبين الجدول التالي عمليات الإسناد الحسابية مع أمثلة وشروح لها.

عملية الإسناد	مثال	الشرح
+=	c+=10	c=c+10
-=	c-=10	c=c-10
*=	c*=10	c=c*10
/=	c/=10	c=c/10
%=	c%=10	c=c%10

عمليات الزيادة بواحد والإنقاص بواحد :

يتوفر أيضاً في لغة c++ عملية الزيادة بواحد الأحادية unary increment operator(++) وعملية الإنقاص بواحد الأحادية unary decrement operator (--) ويلخص الجدول التالي كيفية استعمالهما :

العملية	التسمية	مثال	الشرح

++	عملية الزيادة بواحد أمامية	++a	زيادة قيمة a بواحد ثم استخدام القيمة الجديدة
++	عملية الزيادة بواحد خلفية	a++	زيادة قيمة a بواحد بعد استخدام القيمة القديمة
--	عملية إنقاص بواحد أمامية	--b	إنقاص قيمة b بواحد ثم استخدام القيمة الجديدة
--	عملية إنقاص بواحد خلفية	b--	إنقاص قيمة b بواحد بعد استخدام القيمة القديمة

مثال توضيحي :

```
# include < iostream.h>
main ()
{
        int c ;
        c = 3;
        cout << c << "\n" ;
        cout << c ++ <<"\n" ;
        cout << c << " \n" ;

        c = 3
cout << c << "\n" ;
        cout << c ++ <<"\n" ;
        cout << c << " \n" ;
        return 0;
}
```

وتكون نتائج هذا البرنامج هي:

```
3
3
4
```

3

4
4

مثال :

أكتب برنامج يلخص نتائج امتحان مادة ما لعشرة طلاب وذلك بعد أن أعطيت قائمة بأسماء الطلاب ومقابل كل اسم تم وضع القيمة 1 إذا كان الطالب ناجح والقيمة . إذا كان الطالب راسب في الامتحان

```
# include < iostream.h>
main ( )
{
    int r , p, f ;
    p = 0 ; f = 0 ;
    for ( int i = 1 ; i < = 10 ; i ++ )
    {
        cout << " enter result : "; cin >> r ;
        if ( r = = 1 )
                p + = 1 ;
        else
                f+= 1 ;
    }
    cout << " passed : " <<p <<"\n" ;
    cout << " failed : ' << f << "\n" ;
    return 0 ;
}
```

```
(Intactive C:\TCWIN45\BIN\NONAME03EXE)
        enter result      : 1
```

```
enter result      : 1
enter result      : 1
enter result      : 0
enter result      : 1
enter result      : 0
enter result      : 0
enter result      : 1
enter result      : 1
enter result      : 0
passed : 6
failed    : 4
```

* بنية الاختيار المتعدد switch :

يمكن أن تصادفنا حالة خاصة في إحدى البرنامج تحتوي على سلسلة من القرارات التي تتعلق بنتائج متعدد لفحص قيمة متحول أو تعبير ما ، ويمكن أن تؤدي كل نتيجة من هذه النتائج إلى القيام بفعل مختلف عن الآخر . لذلك توفر لغة C++ البنية switch من أجل التعامل مع حالات اتخاذ القرار المتعلقة بعد اختيارات ، ولها الشكل العام التالي:

```
switch ( expression )
{
  case  constant 1 : statement 1 ;
  case  constant 2 : statement 2 ;
  case  constant 3  : statement 3 ;
  case  constant 4  : statement 4 ;
  .
  .
  case constant n :  statement n ;
  default : statement 0 ;
```

```
}
```

مثال 1:

أكتب برنامج لإعطاء اسم اليوم من أيام الأسبوع عند إعطاء رقمه.

```
# include < iostream.h>
main ()
{
        int c ;
        cout << "enter number : " ;
        cin >> c ;
        switch (c )
{
        case 1 : { cout << " saturday " ; beak ; }
        case 2 : { cout << " sunday " ; beak ; }
        case 3 : { cout << " monday " ; beak ; }
        case 4 : { cout << " tuesday " ; beak ; }
        case 5 : { cout << " wednesday " ; beak ; }
        case 6 : { cout << " thursday " ; beak ; }
        case 7 : { cout << " friday " ; beak ; }
        default : { cout << " that number is out of range " ; }
}
        return 0 ;
}
```

```
(Intactive C:\TCWIN45\BIN\NONAME07.EXE)
        enter number    : 7
        friday
```

مثال2:

أكتب برنامج يقوم بقراءة عددين ومن ثم يعطي ناج جمعها وطرحهما وضربهما مستخدماً لعرض ذلك شاشة خيارات.

```
# include < iostream.h>
main ( )
{
        int  n , x, y ;
        cout << "1:  جمع العددين " ; cout <<"\n";
        cout << "2 :  طرح العددين " : cout <<"\n";
        cout <<3:  ضرب العددين" ; cout <<"\n";
        cout << ******************" ; cout <<"\n";
        cout <<  أدخل العدد الأول " ; cin >>x; cout <<"\n";
        cout <<  أدخل العدد الثاني " ; cin >> y ; cout <<"\n";
        cout <<  أدخل رقم الخيار" ; cin >> n ; cout << "\n";
while ( n!=0)
{
        switch ( n )
        {
        case 1:
                { cout << x+y ; break ; }
        case 2:
                { cout << x-y ; break ; }
        case 3 :
```

```
        ( cout << x*y; break; }
    default :
        { cout << "  الرجاء إدخال أحد أرقام الخيارات المتاحة " ; cin>>n;}
    }
 }
 return 0;
 }
```

أمثالة عامة:

1- أكتب برنامج لقراءة ثلاث أعداد a, b, c ثم التحقق هل تصلح هذه الأضلاع لأن تكون أضلاع مثلث أم لا ، وبمعنى آخر هل يمكن أن نجد مثلث أطوال أضلاعه هي a, b, c .

```
# include < iostream.h>
# include < math.h>   // int abs (int) الملف الرأسي الحاويعلى جميع التواب الرياضية وتم استخدامه
                                من أجل التابع
main ( )
{
 int  a , b, c ;
 cout << " a : " ; cin >> a ;
 cout << " b : " ; cin >> b ;
 cout << " c : " ; cin >> c ;
 if ((a+b>c) && (abs(a-b)<c)&&(b+c>a)&&(abs(b-c)<a) &&(a+c>b) && (abs (a-c)<b))
 cout << " triangle " ;
else
 cout << " not triangle " ;
return 0;
```

```
}
```

```
(Inactive C:\TCWIN45\BIN\NONAME00.EXE)
        a : 5
        b : 4
        c : 3
        triangle
```

2. أكتب برنامج لحساب !n

```
# include < iostream.h>
main ( )
{
        int n ;
        double fact = 1 ;
        cout << " enter value n: " ; cin >> n;
        if ( n = = 0 )
                cout << " n! = 1;
        else
        {
                for ( int i = 1 ; i < = n ; i ++)
                fact * i  ;
                cout << " n ! = " << fact ;
        }
```

```
        return 0 ;
 }
```

```
(Intactive C:\TCWIN45\BIN\NONAME01.EXE)
        enter value n : 5
        n * = 120
```

---

3. برنامج إيجاد قواسم عدد X

الحل:

إذا فرضنا أن العدد 30= x فإنا نختبر الأعداد التي قبل x بحيث إذا كان باقي القسمة عليها يساوي الصفر عندئذ يكون العدد قاسما للعدد x .

```
# include < iostream . h >
main ()
{
  int  x  ;
  cout << " enter number : " ; cin >> x ;
  for ( int i = 1 ; i < =x ; i ++)
  if ( x % i = = 0 )
        cout << i << " \n";
  return 0 ;
}
```

```
(Inactive C:\TCWIN45\BIN\NONAME02.EXE)
        enter number : 30
        1
        2
        3
        5
        6
        10
        15
        30
```

4. برنامج يقوم بقراءة عدد ما x ومن ثم يحدد هل هذا العدد أولي أم لا.

ملاحظة للحل :

1- لا يوجد في لغة الـC++ نمط بولياني لذلك ننشئ نمط من خلال النمط التعدادي enum.

2- لحل هذه المسألة يلزمنا متحول اختبار f من نوع Boolean ففي البداية نسند القيمة false إلى هذا المتحول أي نفرض أن العدد ليس أولي ، ومن ثم نبحث هل هناك عدد يقسم x وفي حال وجوده نسند لـf القيمة True . وفي النهاية نختبر قيمة المتحول f وأعتماداً عليه نحدد هل العدد أولي أم لا.

```
# include < iosteam.h >
enum boolean {true, false };                    // التصريح عن نمط تعدادي
main ( )
{
boolean f = false ;
int x ;
cout <<' enter number: " ; cin >> x ;
```

```
for ( int i = 2 ; i <x ; i ++)
  if ( x % i = = 0 )
  f = true ;
if ( f = = false )
  cout << " the x number is primary " ;
else
  cout << " the x numbe is not primary " ;
return 0 ;
}
```

```
(inactive c:\tcwin45\bin\noname03.exe)
        enter number : 67
        the  x number is primary
```

5. أكتب برنامج لحساب الحدود العشرة الأولى لهذه السلسة :

$$z = 1 - \frac{1}{1} + \frac{1}{2} - \frac{1}{3} + \frac{1}{4} - .....$$

```
# include < iostream.h>
# include < math.h>
main ( )
{
  int  n ;
  float z = 1;
  cout << " enter n: ' ; cin >> n;
  for ( int i = 1 ; i <n ; i ++ )
  if ( i % 2 = = 0 )
```

```
        z+= pow(i , -1) ;              // math تابع الرفع لقوة ويوجد في الملف
 else
        z-=pow (i , -1 ) ;
 cout << " z = " << z ;
 return 0 ;
}
```

```
(Inactive C:\TCWIN45\BIN\NONAME04.EXE)
        enter  n : 15
       z = 0.341295
```

6. أكتب برنامج لإيجاد القاسم المشترك الأعظم لعددين وذلك باستخدام طريقة إقليدس التي تتلخص كما يلي:

أقوم بطرح العدد الأصغر من العدد الأكبر وأجعل حاصل الطرح مكان الأكبر حتى تصبح القيمتين متساويتين فتكون قيمة التساوي هذه هي القاسم المشترك الأعظم GCD .

مثال : العددين 15 و 20

	20	15
	5	15
	5	10
القاسم المشترك الأعظم	5	5

```
#include < iostream.h>
main ( )
{
        int x , y ;
```

```
        cout << "enter x : " ; cin >> x ;
        cout << " enter y : " ; cin >> y;
        while ( x! = y )
        {
if ( x > y )
        x - = y ;
else
        y - = x ;
}
cout << " the gcd is " << x ;
return 0 ;
}
```

(Inactive C:\TCWIN45\BIN\NONAME05.EXE)

enter x : 10

enter y : 35

the gcd is 5

---

7. أكتب برنامج لقراءة n عدد ثم حساب مجموع هذه الأعداد ومتوسطها وأكبر وأصغر عدد فيها: ملاحظة:

دائماً لحساب أكبر أو أصغر عدد من بين مجموعة أعداد ، نفرض أن العدد الأول هو الكبير ثم نختبر باقي الأعداد وكلما ظهر عدد أكبر جديد نجعله هو العدد الأكبر ، وهكذا حتى تنتهي مجموعة الأعداد . (بالنسبة للعدد الأكبر).

```
# include < iostream.h>
main ( )
{
```

```
    int n , x , sum , max , min ;
    cout << " enter n : " ; cin >> n;
    cout << " enter the first number : " ; cin >> x ;
    sum = x ; min = x ; max = x ;
    for ( int i = 2 ; i < = n ; i ++)
    {
        cout << " enter number : " ; cin >> x ;
        sum + = x ;
        if ( x > max ) max = x ;
        if ( x < min ) min = x ;
    }
    cout << " sum is " << sum << "\n" ;
    cout << " avg is " << ( float ) sum /n << "\n" ;
    cout << " max is " << man << " \n" ;
    cout << " min is " << min << "\n" ;
    return 0 ;
}
```

```
(Inactive C:\TCWIN45\BIN\NONAME06.EXE)
    enter  n : 4
    enter the first number : 22
    enter number : 13
    enter number : 24
    enter number : 44
```

```
sum is 103
avg is 25.75
max is 44
min is 13
```

---

8. أكتب برنامج لقراءة عدد ما والتحقق فيما إذا كان عدم تام أم لا .

الحل :

نقول عن عدد ما أنه عدد تام إذا كان مجموع قواسم هذا العدد (ما عدا العدد نفسه) يساوي العدد نفسه .

مثال :

العدد 6 هو عدد تام لأن مجموع قواسم العدد 6 تساوي 06 (1+2+3=6)

```
# include <iostream.h>
main ( )
{
        int x ;
        int sum = 0 ;
        cin>> x ;
        for ( int i = 1 ; i < x ; i ++)
        if ( x % i = = 0 )
                sum + = i ;
        if ( sum = = x )
                cout << " perfect " ;
```

```
        else
                cout << " not perfect " ;
        return 0 ;
 }
```

```
 (Inactive C:\TCWIN45\BIN\NONAME07.EXE)
        enter number = 28
       perfect
```

---

9. أكتب برنامج لإيجاد جميع الأعداد التامة ضمن مجال [1..n]

```
# include < iostream.h>
main ( )
{
  int n , sum = 0 ;
  cin>>n ;
  for ( int i = 1 ; i <=n ; i ++ )
        {
        for ( int j = 1 ; j < i ; j++)
                if ( i % j = = 0 )
                        sum + = j ;
                if ( sum = = i )
                        cout << "  " <<i << endl;
                sum = 0 ;
        }
  return 0;
  }
```

```
(Inactive C:\TCWIN45\BIN\NONAME00.EXE)
      enter  n : 200
     6
     28
```

10. أكتب برنامج لإيجاد المضاعف المشترك الأصغر لعددين:

```
# include < iostream.h>
main ( )
{
      int x , y ;
      cout << " x = " ; cin >> x;
      cout << " y = " ; cin >> y ;
      if ( x > = y)
      {
            for ( int j = x ; j < x ; j++)
                  if ( j % x = = 0 ) && ( j % y = = 0 )
                        { cout <<j ; break ; }
    }
else
   {
      for  ( int j = y ; j < x *y; j++)
            if (( j% x = = 0) && ( j % y = = 0 ))
                  { cout << " " << j ; break ; }
   }
```

```
  return 0 ;
}
```

11. أكتب برنامج لقراءة عددين والتحقق فيما إذا كانا عددين صديقين أم لا .

الحل :

نقول عن عددين أنهما صدقيقين إذا كان مجموع قواسم العدد الأول (ما عدا العدد نفسه) يساوي العدد الثاني والعكس بالعكس.

```
# include<iostream.h>
main ( )
{
   int x , y , i ;
    int sum 1 = 0 , sum 2=0
        cout <<"x="; cin>>x;
        cout <<" y=" ; cin >> y;

        for ( i = 1 ; i < x ; i ++ )
            if ( x % i = = 0 )
                sum 1 + = i ;
        for ( i = 1 ; i < y ; i ++)
             if ( y % i  = = 0 )
                sum 2 + = i ;

        if ( sum 1 = = y && sum 2 = = x )
                cout <<x<<" friend " <<y;
```

```
        else
                cout << x << " not friend " << y;
        reterun 0 ;
}
```

```
(Inactive C:\TCWIN45\BIN\NONAME02.EXE)
        x = 20
       y = 34
       20 not friend  34
```

وظيفة :

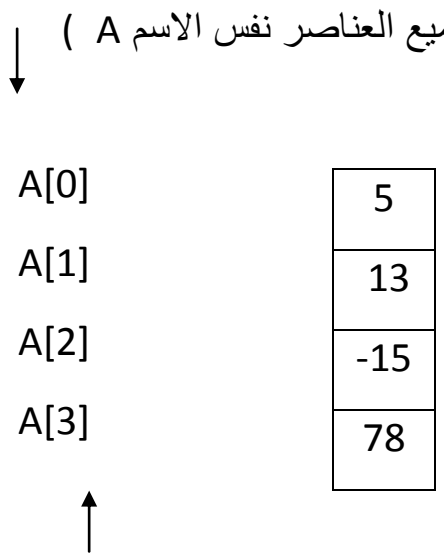
أكتب برنامج لإيجاد جميع الأعداد الصديقة ضمن مجال [1.. n] .

4-المصفوفات

المصفوفات:

المصفوفة هي عبارة عن مجموعة من خانات الذاكرة المتتالية التي لها نفس الاسم ونفس النمط . ومن أجل الرجوع الى خانة معينة من هذه الخانات ضمن المصفوفة ورقم موضع الخانة (العنصر) ضمن المصفوفة وذالك داخل قوسين متوسطين من الشكل ([ ]) .

والشكل التالي يمثل مصفوفة من الإعداد الصحيحة التي أسمها A وهي تتضمن أربعة عناصر .



{رقم موضع العنصر من عناصر المصفوفة (رقم موضع الخانة)}

العنصر الاول من المصفوفة هو دائماً العنصر ذو الرقم صفر وبالتالي يتم الرجوع اليه من المصفوفة A مثلاً على الشكل التالي A[0] وبشكل عام نستطيع القول أننا نرجع الى العنصر ذو الرقم I بكتابة [I-1] اسم المصفوفة

نسمي رقم الموضع الذي نضعه ضمن قوسين متوسطين بالدليل Subscript ويجب أن يكون الدليل عبارة عن عدد صحيح أو تعبير يعطي قيمة صحيحة حيث يتم حساب قيمة التعبير أولاً من أجل تحديد المطلوب ، على سبيل المثال c=2 وb=3 وبالتالي يكون العنصر A[b+c] يمثل العنصر A[5] .

- **التصريح عن المصفوفات :**

تشغل المصفوفات أجزاء محددة من الذاكرة لذلك نقوم بتحديد نمط عناصر المصفوفة وعددها الى المترجم الذي يقوم بدوره بحجز الحجم المناسب في الذاكرة . وبتصريح الشكل العام التالي:

[عدد عناصر المصفوفة] اسم المصفوفة نمط معطيات المصفوفة

مثلاً : int A[5]

يمكن حجز أمكنة لعدة مصفوفات باستخدام تصريح وحيد ، فعلى سبيل المثال ; Y[13] , int x[10]

ويمكن التصريح عن المصفوفات تحتوي معطيات من أنماط أخرى مثل ; float x[100] ، ; char y [100] ، .....

ويسمى هذا النوع من المصفوفات بالمصفوفات ذات البعد الواحد .

- **أمثلة عن طرق اعطاء قيم ابتدائية لعناصر المصفوفة :**

1. يمكن اعطاء قيمة ثابتة لكامل العناصر المصفوفة فعلى سبيل المثال نعطي الصفر لكامل عناصر المصفوفة على الشكل التالي int A[10]={0}.

2. يمكن اعطاء قيم ابتدائية لعناصر المصفوفة أثناء التصريح عنها مثلا ; int A[5]={10,2,34,6,18}

3. يمكن أعطاء قيم ابتدائية لعناصر المصفوفة بالشكل التالي :

```
# include < iostream.h >
main()
{
int a[5]
for(int i=; i <5 ;i ++ )
a[i] = 0 ;
return 0 ;
}
```

مثال 1 :

أكتب برنامج يقوم بطباعة عناصر مصفوفة .

```
# include < iostream.h >
main()
{
int a[5] = {10,2,12,30,67} ;
for (int i =0; i<5 ;i++)
     cout <<"a["<<i<<"] = "<<a[i]<<"\ n ";
return;
}
```

ملاحظات :

1 ـ يسبب التصريح التالي :

```
int [5] ={1,2,34,56,24,14};
```

خطأً قواعدياً لاننا أعطينا ستة قيم لمصفوفة مؤلفة من خمس عناصر فقط .

2 ـ يسبب التصريح التالي :

```
int n[5]={1,2,9,5};
```

اعطاء قيمة الصفر للعنصر الخامس من قبل المترجم .

3 ـ اذا تم حذف حجم المصفوفة أثناء التصريح عنها فان عدد عناصر هذه المصفوفة يصبح مساوياً لعدد القيم الابتدائية المعطاة ضمن القائمة الملحقة بالتصريح . لذلك يقوم التصريح التالي :

```
int n[ ] = {1,2,3,4,5,6} ;
```

بخلق مصفوفة مؤلفة من ستة عناصر .

التصريح عن متحول ثابت :

يكون الشكل العام للتصريح عن المتحول ثابت كالتالي :

```
Const نوع المعطيات اسم المتحول = القيمة ;
```

مثال :

```
Const int size=10;
```

يفيد السطر السابق في التصريح عن متحول ثابت Size وذلك باستخدام الكلمة المحجوزة const 10

ملاحظة هامة :

يجب اعطاء قيمة ابتدائية للمتحولات الثابتة عند التصريح عنها **ولا يمكن تغيير هذه القيمة بعد ذلك** ، تسمى المتحولات الثابتة أيضاً بالثوابت constants .

ومن **الاخطاء البرمجية الشائعة** اعطاء قيمة لثابت من خلال تعليمة تنفيذية مثل :

```
main ( )
{
   const int n ;
  n = 9 ;
   return 0 ;
}
```

وبالتالي تعطي التعليمات السابقة خطأً قواعدياً نتيجة اسناد متحول ثابت ، ويكون التصحيح كما يلي :

```
# includde < iostream. h>
main ()
{
   const int n = 9
   cout <<" the value of constant is : "<< n ;
   return 0 ;
}
```

- **المصفوفات والثوابت :**

يمكن وضع المتحولات الثابتة في أي مكان يمكن أن نضع فية تعبيراً ثابتاً ، فمثلاً يمكن استخدامها في تحديد حجم المصفوفة .

مثال :

```
contest int size = 10 ;

int s [size ] ;
```

تفيد التعليمات السابقة في تحديد حجم مصفوفة S باستخدام الثابت SIZE .

ويفيد استخدام المتحولات الثابتة لتحديد حجم المصفوفات في جعل البرامج أكثر قابلية لتغيير الحجم . فمثلاً حلقة FOR تقوم بتعبئة 10 عناصر يمكن تعديلها لتقوم بتعبئة 1000 عنصر وذلك بتغيير قيمة الثابت المرتبطة به أما في حالة عدم أستخدام الثوابت فيتطلب التعديل السابق عدة تعديلات في أماكن مختلفة من البرنامج .

مثال 1:

اكتب برنامج لطباعة عناصر مصفوفة .

```
# include< iostream.h>
main( )
{
  const int arrasize = 10 ;
  int a [arrasize];
  for (int i=0 ;i<arrasize ; i++)
  {
    a[i]=2+2*i;
```

```
    cout<<a[i]<<"\n";
  }
   return 0;
}
```

مثال 2 :

أكتب برنامج لحساب مجموع عناصر مصفوفة .

```
# include< iostream.h>
main( )
{
  const int arrasize = 10 ;
   int a [arrasize]={1,12,5,4,8,9,7,32,65,91};
   int sum =0;
   for (int I =0;i< arrasize ; i++)
       sum +=a[i];
   cout << "sum = "<<sum;
   return 0;
 }
```

- **مصفوفات الحروف:**

سوف نتعرض الان الى تخزين سلاسل الحروف فى مصفوفات من النمط Char حيث أن أي سلسلة حروف مثلا السلسلة "first" هى فى الواقع عبارة عن مصفوفة حروف . يمكن إعطاء

قيمة ابتدائية لمصفوفة حروف باستخدام سلاسل الحروف فعلى سبيل المثال يقوم التصريح بالشكل التالى

Char str 1[ ] =" first "

بإعطاء قيم ابتدائية لكل عنصر من عناصر المصفوفه str1 حيث يقابل كل منها احد حروف السلسلة "first" ويتحدد عدد عناصر المصفوفة str1 بواسطة المترجم وذلك حسب طول السلسلة المعطاة . من المهم أن نلاحظ أن السلسلة "first" تحتوى على خمسة حروف إضافة الى حرف خاص يحدد نهاية السلسلة وهو الحرف الصفري null character لذلك تتألف المصفوفة str1 من ستة عناصر ويتمم تمثيل الحرف الصفري على الشمل '0 \'. وهذا يعنى أن كافة الحروف تنتهى بالحرف الصغرى ويتم بالتالى التصريح عن المصفوفات التى تتعامل مع هذه السلاسل بحيث تكون ذات حجم كافي لتخظين حروفها إضافة الى الحرف الصفري

يمكن أيضا إعطاء قيم ابتدائية لمصفوفات الحروف باستخدام ثوابت الحروف المفردة ضمن قائمة للقيم الابتدائية . فمثلا يمكن كتابة التصريح السابق على الشكل التالى

```
char str1 [ ] ={' f ',' i ',' r ',' s ',' t ','\0'};
```

وعلى اعتبار ان سلاسل الحروف هي عبارة عن مصفوفات للحروف فيمكن الوصول الى كل حرف من حروفها بشكل منفصل مباشرة باستخدام ذليل عناصر المصفوفة فعلى سبيل المثال يمثل العنصر str1[0] الحرف f' ويمثل الحرف t' العنصر str1 [4]

يمكن ايضا إدخال السلاسل مباشرة الى مصفوفات الحروف باستخدام لوحة المفاتيح وذلك بواسطة cin >> فمثلا التصريح التالى

Char str2 [10] ;

يقوم بإنشاء مصفوفة حروف قادرة عل تخزين سلسلة من 9 أحرف والحرف الصفري ايضا . وتمكن التعليمة التالية :

```
cin >>Str2;
```

على قراءة سلسلة من الحروف من لوحة المفاتيح وتخزينها فى str 2 أما التعليمة التالية

```
cout >> Str2 ;
```

فتساعد على طباعة المصفوفة str2

ملاحظة:

عند قراءة سلسلة حروف من لوحة المفاتيح لم يتم كتابة حجم المصفوفة وإنما فقط إسمها وبالتالى فى حالة عدم التزويد بمصفوفة ذات حجم كافي لاستيعاب الحروف المدخلة من قبل المستخدم بواسطة لوحة المفاتيح تؤدى الى ضياع فى معطيات البرنامج بالاضافة الى اخطاء التنفيذ علما أن cin يقوم بقراءة الحروف المدخلة حتى يصل الى فراغ ولا يهتم بحجم المصفوفة وكذلك الطباعة coutلاتهتم بحجم المصفوفة ويتم طباعة الحروف حتى الوصول الى الحرف الصفري .

مثال توضيحى :

```
# include <iostream.h >
main ( )
{
char str1[10],str2[]="first program";
cin>>str1;
cout<<"str1:"<<str1<<"\n"<<"str2:"<<str2<<"\n";
for(int i=0;str[i]!="\0";i++)
cout<<str[i]<<" ";
Return 0;
}
```

hello there

str 1 : hello

str 2 : first program

h e l l o

فرز المصفوفات :

تعتبر عملية فرز المعطيات (أي وضعها حسب ترتيب معين تصاعدي أو تنازلى مثلا) من أهم التطبيقات الحسابية وبالتالى سوف نقوم بشرح طريقة فرز تدعى بالفرز الفقاعي bubble sort او الفرز بالغوص sinking cort وذلك لان القيم الصغيرة تقوم تدريجيا بشق طريقها تصاعديا الى قمة المصفوفة بينما تقوم القيم الكبيرة بالغوص الى اسفل المصفوفة وتعتمد هذه الطريقة في الفرز على القيام بأكثر من مرور على العناصر وفى كل مرة يتم مقارنة زوجين متتالين من عناصر المصفوفة إذا كان هذان الزوجان مرتبين تصاعديا (أو لهما نفس القيمة) فأننا ندعهما على حالهما وإذا كان مرتين تنازليا فإننا نقوم بالمبادلة بينعما ضمن المصفوفة

يقوم البرنامج التالي بمقارنة العنصرين a[0] و a [1] ثم العنصرين a[1] و a [2] وهكذا حتى نهاية المصفوفه بمقارنة العنصرين a [8] و a [9] وعلى اعتبار أن المصفوفة تحتوى على عشرة عناصر فالبرنامج يقوم بتسع مقارنات تشق خلالها القيمة الكبرى طريقها الى الاسفل بينما تصعد القيمة الصغرى مكانا واحدا وهذا يعنى أن القيمة الكبرى سوف تصل الى الموضع a [9] بعد نهاية المرور الاول أما القيمة الكبرى الثانية سوف تصل الى الموضع

a[8] بعد نهاية المرور الثاني وهمذا حتى المرور التاسع حيث توضع القيمة التاسعة فى الموضع a[1] ويؤدى ذلك لبقاء القيمة الصغرى فى الموضع a[0] إذا نحتاج الى تسعة مرورات لفرز مصفوفة مولفة من عشر عناصر

تتم عملية الفرز من خلال بنية التكرار for المتداخلة وتجرى عملية المبادلة بين العناصر وفقا للتعليمات التالية

hold = a [i] ؛

```
a [i]  = a [i+ l] ;

a [i+ l] = hold ;
```

ونستخدم المتحول الاضافي Hold لتخزين إحدى القيمتين المراد مبادلتهما مؤقتا

```
a [ i ] = a [i+1];

a[ i +1 ] = a[ i];
```

فإذا كانت القيمة a[i] تساوى 10 وقيمة a[i+l] تساوى 8 فإن التعليمة الاولى تجعل قيمة العنصرين مساوية للقيمة العنصرين مساوية للقيمة 8 مما يسبب ضياعا للقيمة 10

```
#include < iomanip.h >
main ( )
{
const int size = 10 ;
int a[ size] ; int hold ;
for ( int i = 0; i< size ; i++)
{
cout << setw (5) << '' a['' << i <<''] ='' ;
cin >> a[ i] ;
cont << endi;
}
for ( int pass = 1 ; pass < size – pass ; i++ )
```

```
if ( a[ i ]> a [ i+ 1 ])
{
hold = a [ i ] ;
a [ i ] =a [ i+1 ] ;
 a[ i+1 ] = hold ;
}
for ( i=0 ; i < size ;i + + )
 cout < < setw (4) < < a[i] ;
return 0;
}
```

ملاحظة :

يتميز الفرز الفقاعي بسهولة البرمجة ولكنة أسلوب فرز بطيء وخصوصا مع المصفوفات الكبيرة

- المصفوفات المتعددة الأبعاد :

يمكن للمصفوفات فى لغة ++cأن تأخذ عدة أبعاد (بعدين وأكثر وصولا إلى 12 دليلا 9 ومن بين الاستخدامات الشائعة المصفوفات الثنائية أو الجداول التي تتألف من الأسطر والاعدمدة . وبالتالي للحصول على عنصر ما من بين العناصر يجب أن نحدد الدليلين : رقم السطر ورقم العمود الذي ينتمي لها العنصر . فمثلا إذا كان لدينا مصفوفة a مؤلفة من ثلاثة أسطر و أربعة أعمدة أي مصفوفة 3X4 فإننا نحدد كل عنصر من عناصر المصفوفة

بـ a [i] [j] حيث أن a اسم المصفوفة و i ، j هما الدليلان المحددان للعنصر المطلوب . حيث تأخذ عناصر السطر الأول القيمة صفر للدليل i أما عناصر العمود الأول فتأخذ

القيمة صفر للدليل j والتالي يمثل a [0] [0] العنصر الأول من السطر الأول والعمود الأول .

صفر إعطاء قيم ابتدائية لعناصر المصفوفة المتعدد الأبعاد بنفس أسلوب المصفوفات ذات البعد فمثلا يمكن إعطاء قيم ابتدائية للمصفوفة a [2] [2] بالشكل التالي

```
int  a [2] [2] ={ {2.4} ,{5.9} };
```

حيث يتم تجميع عناصر كل سطر ضمن قوسين كبيرين . مما يدب على أن القيم 2 و 4 هي قيم العنصرين a [0] [0] و a [1] [0] والقيم 5 و 9 هي a [1] [0] a [1] [1]

```
# include < iostream.h >
# include< iomanip.h >
main( )
{
const int  size 1 = 3 ;
const int  size 2=2;
int a [ size 1 ] [ size 2] ;
for ( int i=0 ; i < size 1 ; ++ )
for ( intj=0 ; j < size 2 ;j ++ )
{
cout < < setw ( 5 ) < < "a["< <" ]["< <j< <"]=";
cin > > a[ i ] { j ] ;
cout < < endl ;
}
```

retum0 ;

في حالة كانت القيم الابتدائية غير كافية لعناصر السطر فانه يتم إعطاء القيمة صفر لبقية العناصر .

مثال : int a [2] [2] ={ {3} ,{4.6} };

يعطى العنصر [0] [0] a القيمة 3 [1] [0] a القيمة 6 أما العنصر[1] [0] a فتسند له قيمة الصفر من قبل المترجم

مثال

اكتب برنامج لقراءة عناصر مصفوفة ثنائية مدخلة من قبل المستخدم

تمارين عامة :

1- اكتب برنامج لقراءة صف ذو بعد واحد ثم طباعته على الشا شة

```
#inlude <iostream.h>
#inlude <iomanip.h>
main ( )
{
const int size =3;
```

```
int a[size ];
for ( int I = 0 : < size : i++ )
for (i=0;<size ;i++)
cout <<''a[''<<''a[''<<i<<'']=''<<a[i] <<setw(5);
retum 0;
}
```

2- اكتب برنامج لقراءة قيم صف ذو بعد واحد ثم احسب مجموع ومتوسط عناصر هذا الصف بالإضافة إلى اكبر واصغر عنصر

```
#inlude <iostream.h>
main ( )
const int size =3;
for ( int I = 0 : < size : i++ )
    cout <<''a[''<<i<<'']= ;
cin>>a[i];
}
int sum= 0,max =a[0],min=a[0];
for(i=0;i<size;i++)
{
```

```
sum+=a[i];
if(max<a[i])max=a[i];
if(min>a[i])min=a[i];
}
cout<<''sum is:''<<sum<< end1;
cout<<average is: ''<< sum/size << endl;
cout <<''max is:''<<max<<end1;
cout<<''min is :''<<min <<end1;
```

3- اكتب برنامج لقراءة عناصر صفين بعد أحادي ثم احسب مجموع هذين الصفين وحداؤهما

```
#inlude <iostream.h>
#inlude <iomanip.h>
main ( )
const int size =3;
{
int a[size],b[size],c[size];
{
cout<<''a[''<<i<<]='';
```

```
cin>>a[i];
}
for(i=0;i<size;i++)
{
cout <<"b["<<i<<"]=";
cin>>b[i];
}
for(i=0;i<size;i++)
{
c[i]=a[i]+b[i];
cout << setw(10) <<"c["<<"]="<<c[i];
mul+=a[i]*b[i];
cout<<setw(15)<<"mul is:"<<end1;
retum 0;
```

4- اكتب برنامج لقراءة قيم مصفوفة ذات بعدين ثم أطبع هذه القيم حسب الشكل الرياضي المتعارف علية

مثال :

9	5	1
3	7	4
2	6	8

```
#inlude <iostream.h>
#inlude <iomanip.h>
main ( )
const int size 1=3;
const int size 2=4;
int a[size1][size2];
for(int i=0 ;i<size1;i++)
   for(int j =0;j<size2;j++)
{
  cout <<''a[''<<i<<''][''<<j<<'']='';
cin>>a[i][j];
}

for(int j=0; j<size2 ;j++)
     cout << setw(5) <<a[i][j];
cout<<end1;
}
return 0;
}
```

5- أكتب برنامج لقراءة قيم مصفوفة مربعة ثم احسب مجموع عناصر القطر الرئيسي ومجموع عناصر القطر الثانوي

ملاحظة :

- عناصر القطر الرئيسي هي [j] [i] a حيث j = I

- عناصر القطر الثانوي [j] [i] a حيث n- 1 = i+ j ، n بعد المصفوفة .

الحل:

```
#inlude <iostream.h>
const int size =4;
main ( )
{
        int b [size][size];
        int I,j,sum1=0,sum2=0;
        for (i=0;i<size;i++)
        for(j=0;j<size;j++)
{
cout<<'' sum master : ''<< sum1<<end1;
cout<<''sum primary :''<<sum2<<end1;
return 0;
}
```

6- اكتب برنامج لقراءة قيم مصفوفة ذات بعدين ومن ثم قراءة قيمة عددية ما والتحقق من وجودها ضمن قيم المصفوفة أم لا .

```
#inlude <iostream.h>
const int size =3;
enum bool {true, false};
main( )
                    int b [size][size]; int I,j,x,
                    bool f =false ;
                   for(i=0;i<size;i++)
                      for(j=0;<size;j++)
                       {
                       cout <<''b[''<<i<<''][''<<j<<'']='';
                        cin >>b[i][i];
```

7- أكتب برنامج لحساب منقول مصفوفة ذات بعدين

```
#inlude <iostream.h>
#inlude <iomanip.h>
main ( )
{
const int size =3;
```

```
int b [size][size],c[size][size];int i , j ;
for(i=o ;i<size;i++)
for(j=0;<size;j++)
{
cout<< '' b [''<<i<<''][''<<j<<'']='';
cin>>b[i][j];
}
for(i=o;i<size;i++)
   for(j=o;j<size;j++)
      c[i][j]=b[j][i];
for(i=o;<size;i++)
```

برنامج لحساب منقول مصفوفة ذات بعدين

```
for(j[O;j<size;j++)
   cout <<c[i][j];<<setw (5);
cout <<end1;
}
return 0;
```

8- اكتب برنامج لعكس قيم صف a ذو بعد واحد جديد b أي يصبح أول عنصر من a آخر عنصر من b وهكذا .

مثال a [1 , 5, 9 ,4 , 7] ⟶ b [7, 4 , 9 , 5 , 1]

```
#inlude <iostream.h>
#inlude <iomanip.h>
main ( )
{
const int n =5;
int a[n], b [n];
for(int i=o;i<n;i++)
{
  cout <<''a[''<<i<<'']='';
cin>>a[i];
}
for(i=o;<n;i++)
{
   b[i]=a[ (n-1) –i];
cout <<''b[''<<i<<'']=''<<b[i] <<setw(5);
}
return 0;
```

9- أكتب برنامج للتحقق من تناظر مصفوفة مربعة .

```
#inlude <iostream.h>
#inlude <iomanip.h>
main ( )
{
const int n =3;
int a[n] [n];
 bool f=true;
for(inti=0;i<n;i++)
   for(int j=0; j;<n;j++)
{
   cout <<''a[''<<i<<''][''<<j<<'']='';
         }
    for( i0;<n;i++)
        for(int j=0; j;<n;j++)
            if(a[i][j]!=a[j][i])
                  f=false
    if(f= = true)
        cout <<''mathed'';
else
}
for (i=0;i<n;i++)
  for(int j=0; j<n ;j++)
```

```
        if(a[i][j]!a[i][j]
            f=false
if( f = = true )
    cout << ''mathed'';
else
   cout<<''no mathed'';
return 0;
}
```

10- نقول عن جملة أو عدد انه palindrome إذا أمكن قراءتها من البداية الى النهاية وبالعكس

مثال 12321 - 555 - 45554 - 121 - radar .....

أكتب برنامج يقوم بإدخال سلسلة (من الحروف أو من الأعداد الصحية) مؤلفة من خمس خانات كحد أقصى ويتحقق فيما إذا كان هذا العدد هو palindrome أو لا.

```
#inlude <iostream.h>
#inlude <iomanip.h>
main ( )
{
   const int n=5;
   char s [n];
   int i= o ;bool f = true;
   cin >>s;
```

```
   while(s[i]!=' 0')
{
   if(s[i]!=s[(n-1)-i])
        f=false;
    i=i+1;
}
if(f = = false )
    cout <<'' not palindrome';
else
    cout <<''palindrome '';
return0;
}
```