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 السلسلة :

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

```
# 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] .