البرمجـــــــة بلغـــــة سي C++

تعتبر لغة C++ من أشهر اللغات التي تتمتع بطابع القوة والمرونة لإنتاج أسرع برامج وأفضلها أداءاً .وعلى الرغم من وجود العديد من لغات البرمجة الأخرى إلا أنها تفتقر شمولية لغة C++ وقوتها . فاللغة C++ تتميز بقابليتها على معالجة التطبيقات الكبيرة والمعقدة، والقوة في صيانة البرامج المكتوبة بها مما يوفر وقتاً في تصميم البرامج وتطويرها.

الخطوة الاولـــى :

سوف تركز هذه الوحدة على إفهامك أساسيات لغة السي بلس بلس؛ ولتعلم أن أفضل طريقة لتعلم أي لغة برمجية هي البدأ فوراً بكتابة أكوادها ، لذلك ابدأ بكتابة الكود الاول التالي:

```
1. # include <iostream.h>
2. main()
3. {
4. cout << "Hii C++ " ;
5. }
```

الكود أعلاه يطبع لك الجملة Hii C++ دعنا نقوم الآن بشرح الكود. السابق.

السطر الأول:

هذا السطر يعتبر أحد أهم الأسطر والتي قلما تجد برنامج لا يتضمن مثل هذا السطر . هذا السطر يخبر المترجم بأن يقوم بتضمين المكتبة iostream في البرنامج ، والمكتبة iostream هي التي تقوم بعمليات الإدخال والإخراج في برامج السي بلس بلس؛ حتى تفهم كيف ننطق مثل هذا السطر فإن # تنطق باوند أو هاش وهي تعني موجه ثم كلمة include والتي تعني تضمين ثم نلفظ المكتبة iostream وهي في الأساس اختصار للجملة input output stream أي أن السطر الأول يقوم بتوجيه المترجم ليقوم بتضمين المكتبة iostream في البرنامج

السطر الثاني والثالث والخامس:

هذا ما يعرف بالتابع أو الدالة main( ) وجميع البرامج في السي بلس بلس وحتى البرامج المتقدمة جداً جداً يجب أن تكون فيها هذه الدالة main( ) تستطيع أنت أن تقوم بكتابة دوال أخرى غير ال main( ) لكن البرنامج لن يعمل إلا بوجود هذه الدالة فهي اللب الأساسي لأي برنامج وكما تلاحظ فإن الدالة main( ) تبدأ بقوس يفتح في السطر الثالث وتنتهي بقوس إغلاق في السطرالخامس، بينما جميع العبارات والجمل والأوامر التي بين قوس الإغلاق والفتح هي جسم الدالة main( ) وبالطبع فلن يمكنك أن تقوم بكتابة أوامر خارج ما يحتويه هذين القوسين.

السطر الرابع:

في السطر الأول قمنا بالطلب من المترجم أن يقوم بتضمين المكتبة iostream , إحدى الخدمات التي تقدمها هذه المكتبة هو الكائن cout ,

الكائن cout يختص بالمخرجات ، أي إذا أردت إخراج أي كتابات على الشاشة فيجب عليك كتابة هذه الكلمة cout بعد ذلك قمنا بكتابة حرفين غريبين قليلاً ألا وهما >> ، في الحقيقة فهذين ليسا حرفان بل هما معامل ، مثله مثل عملية الجمع أو الطرح ويسمى معامل الإخراج حيث يقوم بعمليات الإخراج أي أن جميع ما ستكتبه لاحقاً سيقوم الكائن cout بإخراجه .بعد ذلك كتبنا الجملة المراد إخراجها ألا وهي Hii C++ . عليك أن تنتبه إلى أن الجملة المطبوعة على الشاشة بين علامتي تنصيص هكذا ("Hii C++") بعد ذلك وضعنا العلامة الفاصلة المنقوطة ; لنخبر المترجم أن الأمر انتهى وعليه أن يذهب إلى الأمر التالي.

الخطوة الثانية:

بالنسبة للخطوة الثانية فهذه المرة سنقوم بكتابة كود بسيط ولكنه متقدم بالنسبة لأي مبتدئ برمجة ألا وهو عبارة عن كود يقوم بجمع عددين تقوم أنت بإدخالهما.

```
1. # include <iostream.h>
2. main()
3. {
4. int num1 , num2;
5. cout << "the first number:\n " ;
6. cin >> num1;
7. cout << " the second number:\n";
8. cin >> num2;
9. cout << "the Value is: " << num1+num2;
10. }
```

السطر الرابع:

كما قلنا فالمطلوب أن يقوم مستخدم البرنامج بإدخال عددين اثنين ، ألا تلاحظ معي أن هذان العددان في لغة الرياضيات هما متغيران اثنين ، الامر نفسه بالنسبة للبرمجة فعلينا أولاً اعتبار هذان العددان متغيران وبالتالي نطلب من البرنامج أن يقوم بحجز ذاكرة لعددين اثنين ثم إذا قام مستخدم البرنامج بإدخال عددين فإن البرنامج يقوم بأخذ العددين وتخزينهما في موقع الذاكرة الذي طلبنا من البرنامج حجزهما في البداية ، وهذا واضح في السطر الرابع فلقد قمنا بتسمية متغيران اثنين الأول هو num1والثاني هو num2 الآن كيف يعلم البرنامج أن num1 و num2 هما عددان بإمكانه فعل ذلك عن طريق أول كلمة في السطر الرابع ألا وهي int وهي إختصار للكلمة integer أي الأعداد الصحيحة والاختصار int . لاحظ أيضاً أن

هناك فاصلة غير منقوطة , بين اسمي المتغيران وهذه ضرورية فكيف يعرف البرنامج أنك انتهيت من كتابة اسم المتغير الأول ، ولاحظ معي أيضاً أن الأمر انتهى بالفاصلة المنقوطة (;)

الآن هناك ملاحظة جديرة بالاهتمام وهي أنه بإمكانك تعديل السطر الرابع ليصبح سطران اثنين هكذا:

```
1. int num1 ;
2. int num2;
```

والطريقتين صحيحتان إلا أن الطريقة الأولى أفضل بسبب أنها مختصرة.

السطر الخامس والسابع:

السطران الخامس والسابع في أغلبهما مفهومان فلا جديد فيهما إذا لم تفهمهما فارجع إلى فقرة الخطوة الأولى ؛ إلا أن هناك أمراً بالغ الأهمية؛ لاحظ معي الجملة التي طلبنا من البرنامج طباعتها:

```
"the first number:\n "
```

كما ترى فإن السبب في أننا طبعنا هذه الجملة والجملة في السطر السابع حتى نوضح لمستخدم البرنامج أن عليه إدخال العدد الأول أو العدد الثاني حسب السطر السابع ؛ ولكن هل ترى آخر الجملة السابقة أقصد هذه العلامة **"\n"** إن هذه العلامة لن يقوم البرنامج بطباعتها بل إن هذه العلامة في الحقيقة إختصار ، فهذه العلامة تطلب من مؤشر الكتابة أن يذهب إلى سطر جديد وبالتالي فحينما يقوم مستخدم البرنامج بإدخال العدد الاول فلن يقوم بإدخاله بجانب الجملة السابقة بل في السطر التالي من الجملة السابقة.

السطر السادس والثامن:

بعكس السطران الخامس والسابع فإن هذان السطران يطلبان منك إدخال عددين اثنين ، حيث يقوم المترجم بأخذ العدد الذي تقوم بإدخاله في السطر السادسويضعه في المتغير **num1** ويأخذ العدد الذي تقوم بإدخاله في السطر الثامن ويضعه في المتغير **num2** هذه هي الفكرة ، فهناك كائن جديد يختصبالإدخال هو **cin** بعد ذلك نستخدم معامل الإدراج وهو هكذا **<<** وليس معامل الإخراج الخاص بالكائن **cout** ثم نكتب اسم المتغير الذي نريد من المستخدم أن يقوم بإدخال قيمة هذا المتغير.

السطر التاسع:

يقوم الكائن **cout** أيضاً بطباعة المتغيرات ، وفي نهاية الجملة المطبوعة يقوم البرنامج بطباعة هذه العبارة **num1+num2** وبما أنها ليست بين علامتي تنصيص فلن يقوم البرنامج بطباعتها كجملة عادية على الشاشة أي هكذا **num1+num2** بل سيقوم بأخذ قيمة المتغير **num1** ويجمعها مع قيمةالمتغير **num2** ويطبع الناتج.

حاول كتابة الكود السابق وتجريبه على جهازك.

القراءة (الإدخال) والكتابة:

بإمكانك الطلب من البرنامج طبع أي قيمة على الشاشة بواسطة الكائن **cout** ، وبإمكان هذا الكائن طباعة أي قيمة عبر معامل الإخراج >> ، وبإمكانه طباعة المتغيرات أو الجمل التي أنت تريد إظهارها ولكي تظهر جمل على الشاشة فعليك كتابتها بين علامتي تنصيص، كما في هذا المثال:

```
cout << "Hello C++";
```

أما إذا أردت إظهار قيم أحد المتغيرات فعليك كتابة اسمه دون علامتي تنصيص كما هنا:

```
cout << a ;
```

مع العلم أن a عبارة عن متغير. أيضاً فبإمكانك طباعة أكثر من متغير أو جملة دفعة واحدة ،كما في هذا السطر:

```
cout << "Please: " << a << b << "Hello" ;
```

أيضاً هناك عبارة بإمكانك إستخدامها لإفراغ المنطقة الوسيطة من جميع الاحرف العالقة أو بشكل مبتدئ طباعة سطر جديد ، انظر إلى هذا السطر :

```
cout << "Hello" << endl << "World" ;
```

سيكون مخرج هذا الأمر على الشاشة هكذا:

```
Hello
World
```

أيضاً هناك بعض الخصائص للكائن **cout** وهي سلاسل الإفلات ، وقد استخدمنا أحدها في المثالين السابقين وهو **\n** والذي يقوم بطباعة سطر جديد لك.

بعض سلاسل الإفلات:

جدولة أفقية تترك **3** فراغات **\t** .

الإنتقال إلى صفحة جديدة **\n** .

إعادة المؤشر إلى بداية السطر **\r** .

يقوم بإصدار صوت تنبيه **\a** .

الحذف الخلفي **\b** . (**back space**)

سلاسل الإفلات نقوم بكتابتها ضمن الجمل أي بين علامتي التنصيص. " "

بالنسبة للإدخال في السي بلس بلس فبامكانك بواسطة الكائن **cin** وهذا الكائن يستخدم فقط مع المتغيرات وليس شيء أخر ، وقد رأيت بعضاً من استخداماته في الامثلة السابقة.

الأساسيات :

راجع الخطوتان السابقتان وافهمهما جيداً قبل الدخول في هذه الفقرة. بالرغم من أنك لم تقم إلا بخطوتين فقط في سبيل تعلم لغة البرمجة السي بلس بلس إلا أنها قفزة كبيرة ولا شك وعلى الأقل فقد أعطتك تلك الخطوتان مقدمة عامة عن أساسيات البرمجة؛ فلا بد وأنك صادفت الكلمات التالية:

التعابير ، الأنماط ، المتغيرات ، الكتل ، التوابع ، المكتبات القياسية ، العمليات ، كائنات الإدخال والإخراج. لا تقلق فبعض الكلمات السابقة لم أذكرها صراحة فيما سبق ولكن تعرضت لفكرتها ، سنبدأ الآن بشرح هذه الأساسيات.أيضاً تتعرض هذه الأساسيات لبعض المواضيع المتقدمة وليس الغرض هو حشو المادة العلمية بل لمعرفة مقدمة ولو بسيطة عنها لأن أصغر كود يحتاج في بعض الأحيان لتلك المعلومات.

المتحولات أو المتغيرات Variable :

المتغيرات كما رأينا عبارة عن أسماء تحجز مواقع في الذاكرة حتى يتمكن البرنامج من تخزين البيانات فيها .حين ما تقوم بتعريف متغير فلا بد أن تخبر المترجم باسم هذا المتغير ونوع المعلومات التي ستحفظها فيه.حينما تقوم بتحديد نوع المعلومات للمتغير فإن المترجم يحجز له عدداً من البايتات حسب ذلك النوع فمرة تكون بايتاً واحداً ومرة أخرى تكون اثنان ومرة ثمان بايتات.

تسمية المتغيرات:

من الممكن أن يتألف اسم المتغير من أرقام وحروف شريطة أن يكون أول حرف هو حرف عادي وليس رقم ، ولا يسمح بأن يحتوي الاسم على الأحرف اللاتينية أو الرموز مثل ؟ و@ وغيرها ، وتعتبر الشرطة السفلية حرفاً صحيحاً بالإمكان كتابته ضمن اسم المتغير _ ، أيضاً تفرق لغة السي بلس بلس بين المتغيرات ذات الحروف الكبيرة والأخرى ذات الحروف الصغيرة ، وكعادة برمجية جيدة فمن الأفضل أن يكون اسم المتغير اسماً ذا معنى وهذا يسهل عليك الكثير من مهام تطوير الكود وصيانته.

أنماط البيانات وحجومها:

تعرفنا في فقرة الخطوة الثانية على معلومة مهمة للغاية ألا وهي نمط بيانات **int** ولكن لهذا النمط عيب وحيد فهو لا يحتوي على أي علامة عشرية وحتى تستطيع من تمكن المتغيرات على التعامل مع الاعداد العشرية فلا بد أن تغير نمط البيانات إلى **float** ، وإذا أردت أن تغير أيضاً من ذلك لتصبح المتغيرات قادرة على التعامل مع الحروف فلا بد أن تجعل نمطها **char** او **string**

بالنسبة للأعداد الكبيرة جداً فبإمكانك وضع أنماط أخرى مثل **long** و **double** وجميعها صالحة .

النوع	الحجم	ملاحظات
bool	1	صواب أو خطأ
char	1	0 الى 265
string	256	سلسلة حروف
int	2	يحوي الأعداد الصحيحة
float	4	يحوي الأعداد العشرية
double	4	يحوي الأعداد الكبيرة العشرية

ملاحظات على الأنماط الرقمية:

بإمكانك استخدام صفات على الأنماط الأساسية مثل الصفة **short** و **long** مع ال **int** اللتان تطبقان على المتغيرات من النوع **int** :

```
short int number=0;
long int index=0;
```

وبإمكانك إذا ما أردت استخدام هاتين الصفتين الاستغناء نهائياً عن الكلمة **int** كما في هذه السطرين:

```
short number=0;
long index=0;
```

Some examples are:

```
int    i;
char   c;
float  f;
double d;
```

```
int    i, j, k;
char   c, ch;
float  f, salary;
double d;
```

```
int d = 3, f = 5;
byte z = 22;
char x = 'x';
```

---

ex_1.

```
 main ()
{
  int a, b;
  int c;
  float f;

  a = 10;
  b = 20;
  c = a + b;

  cout << c << endl ;

  f = 70.0/3.0;
  cout << f << endl ;
}
```

result:

```
30
23.3333
```

---

ex_2.

```
main ()
{
  int a, b;
  int g;
  a = 100;
  b = 20;
  g = a + b;
  cout << g;
 }
```

result:

120

---

ex_3.

```
// This program adds two numbers and prints their sum.
#include <iostream.h>
  main()
 {
  int a;
  int b;
  int sum;
  cin >> a >> b;
  sum = a + b;
  cout << "The sum of " << a << " and " << b << " is " << sum <<
"\n";

  }
```

result :

The sum of a and b is _____!

الثوابت Constants:

يوجد بعض المتغيرات التي ترغب في عدم تغييرها أبداً وربما حينما يصل البرنامج إلى عدة آلاف من الأسطر الكودية قد لا تستطيع معرفة إن كان هذا المتغير تغير لذلك فستود جعله ثابتاً ، وفي حال تغير لأي ظرف من الظروف قد يكون خطأ منك فسيقوم المترجم بإصدار خطأ ينبهك بذلك ، وحتى تستطيع أن تقول للمترجم أن هذا المتغير ثابت ، لذلك لا تسمح لأحد بتغيرها حتى أنا المترجم فعليك بكتابة كلمة const قبل نمط المتغير هكذا:

```
const int number=14 ;
```

تذكر حينما تقوم بالإعلان عن أن هذا المتغير ثابت فعليك تهيئته بقيمة في نفس الوقت وإلا فلن تستطيع تهيئته بأي قيمة أخرى لأن المترجم يعتبره ثابتاً ولن يسمح لك بتغييره أي أن السطرين التاليين خاطئين:

```
const int number;
number=14;
```

```
ex_4.
#include <iostream.h>

 main()
{
   const int  L = 10;
   const int  W  = 5;
   const char NEWLINE = '\n';
   int area;
   area = L * W;
   cout << area;
   cout << NEWLINE;
}
result :

50
```

الإعلانات والتعاريف Declarations and Definitions :

كثيراً ما ستجد في هذا الكتاب وغيره من كتب البرمجة عبارتي إعلان وتعريف يجب أن تعرف الفرق بينهما. تفرض عليك لغة السي بلس بلس الإعلان أو التصريح عن المتغيرات قبل استخدامها ، أنظر إلى هذا السطر:

```
int number =4;
```

لقد قمت بالإعلان عن أحد المتغيرات ، أما التعريف فهو الذي ينشأ عنه حجز للذاكرة وبالتالي فإن الإعلان السابق هو نفسه تعريف لأنه يصاحبه حجز للذاكرة ، في أغلب المواضيع الإعلان هو نفسه التصريح ولكن تذكر الفرق بينهما لأنه مهم للغاية وخاصة في المواضيع المتقدمة نسبياً كالمؤشرات والكائنات والتوابع وغيرها.

العمليات الحسابية Arithmetic Operations :

في السي بلسبلستوجد خمسعمليات حسابية:

1-عملية الجمع: (+)

2-عملية الطرح: (-)

3عملية الضرب: (*)

4-عملية القسمة: (/)

5-عملية باقي القسمة(%)

جميع هذه العمليات الحسابية بإمكانك القيام بها على المتغيرات العددية، بالنسبة إلى العملية الخامسة فلا يمكنك القيام بها إلا على أعداد من النوع **int** وليس غيره.

عمليات المقارنة أو العلائقية Relation Operator :

في السي بلس بلس توجد عمليات المقارنة حيث بإمكانك مقارنة أعداد مع بعضها البعض أو مقارنة أحرف من النوع **char** ، وهذه هي عمليات المقارنة في السي بلس بلس:

== , <= , >= , < , > , !=

التعابير وعمليات الإسناد Assignment Operator And Expressions :

هناك معامل آخر لم نقم بشرحه في العمليات الحسابية وهو المعامل (=)هذا المعامل يختلف في السي بلس بلس عن نظيره في الرياضيات هذا المعامل يقوم بإسناد المتغير الذي في يمينه إلى الذي في يساره وهو يستخدم مع المتغيرات الحرفية فبإمكانك إسناد متغير حرفي إلى آخر ، كما يظهر في هذا المثال:

```
int a =5;
int b;
b=a;
```

في هذا السطر فإنك تخبر المترجم بالقول له أنه يجب عليه أخذ قيمة (a) ووضعها في المتغير (b) أيضاً هناك عملية إسناد أخرى ، لنفرضأن لدينا متغير هو i وهو عددي ونريد جمعه بالعدد 2 حينها ستقوم بكتابة:

```
i=i+2;
```

توفر لك السي بلس بلس معامل إسناد أسرع من معامل الإسناد = وأكثر اختصاراً هو =+ ، بالتالي سنختصر السطر السابق إلى هذا السطر:

```
i+=2 ;
```

التعابير الشرطية Conditional Expressions :

هل تتذكر المعاملات العلائقية ، ستظهر فائدتها هنا لنفرضأن لدينا ثلاثة متغيرات ، حيث أننا نقوم بكتابة برنامج يقوم بمقارنة أي عددين وحساب الأكبر منهما ، لنفرضأن المتغيرين أو العددين الذي نود مقارنتهما هما a و b أما المتغير الثالث فسيكون max .

```
1 if (a > b )
2 max = a ;
3 if ( b < a)
4 max = b ;
5 if ( b == a)
6 max = a = b;
```

هنا أحد التعابير الشرطية وهو التعبير if يقوم هذا التعبير باختبار التعبير الذي بين القوسين بعده ، وفي حال نجاح التعبير فإنه ينفذ الأوامر التي بعده وفي حال عدم نجاحه فإنه يخرج تلقائياً ولا ينفذ الأوامر التي ضمن الكلمة if انظر إلى السطر الأول ، لنفرضأن المتغير a بالفعل هو أكبر من المتغير b حينها سيتم تنفيذ السطر الثاني أما في حال لم يكن كذلك فلن يتم تنفيذ السطر الثاني وسيواصل البرنامج عمله وينتقل إلى السطر الثالث. انظر أيضاً إلى عملية المقارنة في السطر الخامس وهي ==أي هل يساوي المتغير a المتغير b في حال كانا متساويان فإن السطر السادس سيتم تنفيذه ، انظر أيضاً أننا في حالة المساواة لم نقم بكتابة المعامل= والسبب أن المعامل = كما قلنا سابقاً هو معامل إسناد أي يأخذ القيمة التي على يمينه ويضعها على يساره ولا يقوم بمقارنة أبداً أما المعامل== فيقارن بين القيمتين.

عمليات الإنقاص والإزيادة Increment and Decrement Operators :

سنتعرف الآن على عملية غريبة علينا وهذه العمليتين هي عملية الإزيادة ++وعملية الإنقاص. -- ليس ذلك فحسب بل طريقة كتابة هذه العمليتين قد تختلف ، وهي صيغتين إما أن تكون إحدى هذه العمليتين على يمين المتغير وإما على يساره وتختلف في كلا الحالتين ، حتى تفهم ما أعنيه لنفرضأن لدي متغيران الأول هو a والثاني هو b انظر إلى هذه الأسطر:

```
a = ++b ;
```

إن هذا السطر يخبر المترجم بالقول يا أيها المترجم زد قيمة المتغير b رقماً واحداً أي العدد 1 ثم أسند قيمة المتغير b إلى المتغير a فلو افترضنا أن قيمة المتغير b هي 6 فحينما يقوم البرنامج بتنفيذ السطر السابق فإنه يقوم أولاً بزيادة المتغير b

زيادة واحدة أي تصبح قيمته 7 ثم يسند القيمة إلى المتغير a أي ستصبح قيمة المتغير a أيضاً 7 الآن لو افترضنا أننا قمنا بكتابة صيغة أخرى وهي هكذا:

```
a = b ++ ;
```

ستختلف العملية هنا ، والآن قم بالتركيز فيما سيكتب ، أولاً سيأخذ المترجم قيمة المتغير b أي تغيير ويقوم بإسنادها إلى المتغير a ثم بعد ذلك يقوم بزيادة المتغير b زيادة واحدة ، أي أن هذه الصيغة عكس الصيغة السابقة فلو فرضنا أن قيمة المتغير b هي 6 فأولاً سيأخذ المتغير هذه القيمة ويسندها إلى المتغير a وبالتالي تصبح قيمة المتغير a هي 6 ثم بعد ذلك يقوم المترجم بزيادة المتغير b .أي أنها ستصبح 7 أيضاً نفس الشرح السابق يطبق على عملية الإنقاص --، مع إختلاف العمل الذي تقومان به طبعاً.

مثال 1 : قم بكتابة كود يقوم بعرض الجملة التالية على الشاشة.

```
Hello
I am a programmer
```

كما ترى فإننا هنا لن نستخدم أي متغيرات (المتغيرات تستخدم لتخزين ما نريد تخزينه في الذاكرة) لأننا لن نقوم بتخزين أي شيء بل كل ما علينا فعله هو عرض بعض الجمل على الشاشة ، الآن إلى الكود:

```
1. #include <iostream.h>
2. int main()
4. {
5. cout << "Hello \n I am a programmer " << endl;
6. }
```

كما ترى فلم نستخدم إلا سطراً وحيداً لتنفيذ المطلوب من السؤال أو المثال \n وهو السطر الخامس، انظر في السطر الخامس إلى سلسلة الإفلات كما قلنا تستخدم هذه السلسلة للإنتقال إلى سطر جديد.

Control Flow بنئ التحكم:

لقد أنجزنا بعضاً من الأكواد المفيدة بواسطة القليل من المعرفة في اللغة ؛ إلا أن الأمر لن يستمر مطولاً هكذا ، فماذا لو طلب منك إنشاء برنامج آلة حاسبة متكاملة تقوم بجميع العمليات وليسبعملية واحدة ، أيضاً ماذا لو طلب منك كتابة برنامج يطلب من المستخدم إدخال قيم أكثر من 100 متغير للقيام بعمليات حسابية أو لكتابة قاعدة بيانات ، حينها سيزداد الكود لدرجة مملة للغاية ، من هنا تظهر فائدة بنى التحكم، والتي تسمح لك بالتحكم أكثر في برنامجك.

جمل بنى التحكم:

تقسم جمل بنى التحكم إلى قسمين رئيسيين ؛ هما:

1-جمل إتخاذ القرارات.

2-جمل تنفيذ الحلقات.

وسنتعرض لكلا النوعين بالشرح والتفصيل

جمل إتخاذ القرار:

تفيد جمل اتخاذ القرار كثيراً في الاكواد ، فهي تسمح لك بالسيطرة أكثر على برنامجك ، أيضاً فلو ألقينا نظرة متفحصة للأكواد السابقة فستجد أنه لا يمكنك السماح للمستخدم بالتفاعل مع البرنامج ، انظر إلى برنامج الوورد ، إنه يعطيك خيارات واسعة من خلال شريط الأدوات وليسمثل البرامج التي نكتبها حالياً ، من هنا تكمن أهمية وفائدة جمل اتخاذ القرار ، وتذكر أن هناك جملتين رئيسيتين ؛ هما:

1- الجملة if وتفرعاتها.

2- الجملة switch .

الجملة if :

تأخذ الجملة **if** الصيغة العامة التالية:

```
if (expression) {
statement1;
statment2;
}
```

بإمكاننا الإختصار إلى القول أنه إذا كان الشرط الذي تقوم الجملة **if** بإختباره صحيحاً فقم بتنفيذ الجمل التي بين القوسين وفي حال عدم صحة الإختبار فلا تقم بتنفيذ الجملة **if** وإنما استمر في قراءة البرنامج من بعد كتلة **if** فمثلاً انظر إلى هذا الكود:

```
1- #include <iostream.h>
2- main()
3- {
4- int i=0 ,j=0;
5- cin >> i >> j ;
6- if (i > j )
7- {
8- cout << "The number i is bigger than j" ;
9- }
11- }
```

كما ترى فإن هذا الكود يطلب من المستخدم إدخال رقمين ، يقوم البرنامج بمقارنة هذين الرقمين وفي حال إذا كان الرقم الأول أكبر من الرقم الثاني فإنه يطبع رسالة تخبرك بذلك وفي حال أن العددين متساويين أو أن العدد الثاني هو أكبر فلن يتم تنفيذ السطر 8 لعدم صحة شرط الجملة if .

الجملة if/else:

لا يقوم الكود السابق بفعل أي شيء إذا اختل شرط الجملة if وبالرغم من أنه بإمكاننا كتابة جملة if ثانية في حال مساواة العددين وجملة if ثالثة في حال أن العدد الثاني أكبر ، إلا أن ذلك لا يمنع من وقوع أخطاء ، فمثلاً فإن بعض الأشخاص لن يتوقعوا أبداً أن العددين سيكونان متساويان لذلك فإن الحل الأفضل هو أن يكون هناك جملة أخرى موازية للجملة if تبدأ في العمل في حال عدم نجاح إختبار الشرط في الجملة if الصيغة العامة لهذه الجملة هي كالتالي:

```
if (expression)
 {
statement1 ;
statement2;
}
else {
statement3;
statement4;
}
```

بإمكاننا إختصار هذه الجملة إلى القول :أنه في حال عدم نجاح إختبار الشرط في الجملة if فإن البرنامج سيقوم بتنفيذ الكتلة التي تتبع للعبارة else ، أما في حال نجاح اختبار الشرط في الجملة if فإن البرنامج سيقوم بتنفيذ الكتلة التي تتبع للجملة if ولكنه سيتجاهل الكتلة التي تتبع الجملة else .

الآن سنقوم بإعادة كتابة الكود السابق وهذه المرة سنجعله يتعامل مع الحالات الأخرى.

```
13- #include <iostream.h>
14- int main()
15- {
16- int i=0 ,j=0;
17- cin >> i >> j ;
18- if (i > j )
19- {
20- cout << "The number i is bigger than j" ;
21- }
22- else { cout << "error" ; }
23- }
```

لم يختلف الكود الحالي عن الكود السابق إلا في السطر **22** حينما جعلنا البرنامج يعرض على الشاشة رسالة خطأ للمستخدم في حالة عدم نجاح اختبار الشرط في العبارة **if** .

العبارة **if/else** من الممكن أن نطلق عليها العبارة **if** الثنائية الإتجاه لأن البرنامج يتفرع فيها إلى طريقين أو إلى فرعين بعكس الجملة **if** السابقة فإنها توصف بأنها أحادية الإتجاه لأنها تسلك طريقاً واحداً في حال نجاح الشرط.

الجملة else/if:

من الممكن وصف هذه الجملة بأنها متعددة الإتجاهات ، فهي تسمح لك بسلوك الكثير من الطرق بدلاً من طريق واحد فحسب ، انظر إلى صيغتها العامة.

```
if (expression)
 {
      statement1;
      statement2;
      statement3;
 }
else if (expression)
{
      statment1;
}
```

```
else if (expression)
 {
      statement;
 }
else
{
      statement;
}
```

سنقوم الآن بتطوير الكود السابق ليصبح قادراً على التعامل مع جميع الحالات.

```
2- #include <iostream.h>
3-  main()
4- {
5- int i=0 ,j=0;
6- cin >> i >> j ;
7- if (i > j )
8- {
8- cout << "The number i is bigger than j" ;
9- }
10- else if (j > i)
11- {
12- cout << "The number j is bigger than i" ;
13- }
14- else if ( j==i)
15-{
16- cout << "there is no bigger number" ;}
17- else { cout << "error" ; }
18- }
```

ترى الإختلاف عن الأكود السابقة في هذا الكود في الأسطر **10** إلى **17** وقد أضفنا لهذ الكود جملتين **else if** تقوم الأولى بإختبار ما إذا كان العدد الثاني هو الأكبر ثم تطبع جملة تخبر المستخدم بذلك أما الثانية فهي تقوم بإختبار ما إذا كان العددان متساويان وتطبع جملة تخبر المستخدم بأنه ليسهناك رقم أكبر من الآخر أما العبارة **else** الأخيرة فهي تفيدك في حال وقوع مفاجآت جديدة.

قد تتحاذق وتتساءل عن الفائدة المرجوة من العبارة **else/if** وقد تقوم بتعديل الكود الى التالي .

```
1- if (i > j ) {
2- cout << "The number i is bigger than j" ;
3- }
4- if (j > i) {
5- cout << "The number j is bigger than i" ;
6- }
7- if ( j=i) {
8- cout << "there is no bigger number" ;
9- else { cout << "error" ; }
```

أي أنك ستقوم بالاستغناء عن العبارة **else/if** بالعبارة **if** ، ولهذا الرأي عيوب كثيرة وأخرى شنيعة قد تدمر برنامجك وقد تجعله مضحكاً.

في حال إستخدامنا للعبارة **else/if** فإنه في حالة نجاح أي شرط من شروط العبارة **else/if** فإن البرنامج يخرج من هذا التداخل الحاصل من عبارة **else/if** ولن يقوم بإجراء أي إختبار وهذا له فائدة كبيرة فهو يقلل من المجهود الذي يقوم به الحاسب وبالتالي يحسن نواحي كثيرة من برنامجك ، أما في حال إستخدام العبارة **if** فإنه حتى في حال نجاح أي شرط من شروط العبارة **if** فإن البرنامج سيستمر في إختبار الرقم حتى يخرج نهائياً، وبالتالي فهذا يزيد من المجهود الملقى على الحاسب مع العلم أن هذا المجهود سيفيدك كثيراً إذا ما كنت تعمل على مشروعات كبيرة وليس مثل هذا الكود الصغير.

مثال عملي: سنقوم الآن بكتابة برنامج شبيه ببرنامج الآلة الحاسبة.

```
2- #include <iostream>
3-
4-  main()
5- {
6- float a,b;
7- char x;
8-
9- cout << "Enter Number1:\t" ;
10- cin >> a;
11- cout << "Enter Number2:\t" ;
12- cin >> b;
13- cout << "Enter the operator\t";
14- cin >> x;
15-
16- cout << endl << endl;
17-
18- cout << "Result:\t";
```

```
19-
20-
21- if (x=='+') { cout << a+b ;}
22- else if (x=='-') { cout << a-b;}
23- else if (x=='*') { cout << a*b;}
24- else if (x=='/') { cout << a/b;}
25- else { cout << "Bad Command";}
26-
27- cout << endl;
28-}
```

يطلب البرنامج من المستخدم إدخال العدد الأول ثم العدد الثاني ثم العملية الحسابية التي تريد استخدامها تبدأ عبارات الجملة **else/if** من السطر **21** وتستمر حتى السطر **25** ، حيث تختبر المتغير **x** لترى إن كان يقوم يحتوي على أي من العمليات الحسابية وفي حال ذلك فإنها تطبع القيمة الناتجة وفي حال عدم ذلك فإن التنفيذ سيكون في السطر **25** حيث في حال أدخل المستخدم أي عملية أو حرف أو حتى رقم من غير العمليات الحسابية المعروفة فإن البرنامج يطبع عبارة تنبئك بحدوث خطأ ، بعد ذلك يخرج البرنامج نهائياً

لاحظ أنه إذا كان الحرف **x** عبارة عن عملية حسابية فإن الجملة **else/if** لن تقوم بإجراء العملية الحسابية وتخزينها في متغير بل ستقوم بطباعة القيمة فوراً وإجراء العملية الحسابية عليها في نفس الوقت.

الجملة switch:

إحدى جمل إتخاذ القرارات ، إلا أنها هذه المرة تعتبر جملة **if** متطورة ، حيث أنه ليس هناك أي فرق بينها وبينها الجملة **if** متعددة الإتجاهات ، وتصاغ هذه العملية حسب الصيغة التالية:

```
switch (expression)
 {
   case const-expr: statements ;
   case const-expr: statements ;
   default: statements ;
}
```

بإمكاننا إختصار شرح هذه الصيغة العامة ، إلى أنه بإمكانك أن تكتب المتغير الذي تريد إختباره في مثال الآلة الحاسبة كان المتغير **x** وتكتبه بين قوسين بعد عبارة **switch** بعد ذلك تقوم بكتابة الحالات المتوقعة لهذا المتغير بعد الكلمة الدليلية **x** وفي حال مطابقة إحدى هذه الحالات مع المتغير يتم تنفيذ الجمل التي تختص بتلك الحالة وفي حال عدم موافقة أي منها فبإمكانك كتابة حالة عامة تشبه الجملة **else** في مثال الآلة الحاسبة , قد ترى أن هناك الكثير من التشابه بين الجملة **else/if** والجملة **switch** إلا أن هناك بعض الفروق البسيطة التي قد تكون مؤثرة :

الآن سنقوم بإعادة كتابة مثال الآلة الحاسبة ، ولكن هذه المرة بالعبارة **switch** وسترى الفرق بينها وبين الجملة **else/if :**

```
1- #include <iostream.h>
3-
4-  main()
5- {
6- float a,b;
7- char x;
8-
9- cout << "Enter Number1:\t" ;
10- cin >> a;
11- cout << "Enter Number2:\t" ;
12- cin >> b;
13- cout << "Enter the operator\t";
14- cin >> x;
15-
16- cout << endl << endl;
17-
18- cout << "Result:\t";
19-
20- switch (x) {
21- case '+':
22- cout << a+b ;
23- break;
24- case '-':
25- cout << a-b;
26- break;
27- case '*':
28- cout << a*b;
29- break;
30- case'/':
31- cout << a/b;
32- break;
33- default:
34- cout << "Bad Command";
35- }
36-
37- cout << endl;
38-
39- return 0;
40- }
```

هذا هو البرنامج بشكل عام وبالنظر إلى أني قمت بشرحه سابقاً فسأقوم بشرح عبارة **switch ، انظر :**

```
1- switch (x) {
```

```
2- case '+':
3- cout << a+b ;
4- break;
5- case '-':
6- cout << a-b;
7- break;
8- case '*':
9- cout << a*b;
10- break;
11- case'/':
12- cout << a/b;
13- break;
14- default:
15- cout << "Bad Command";
16- }
```

أول شيء يجب النظر إليه أن تفرعات الجملة **switch** ليست مثل جمل **if** السابقة بل تبدأ بالكلمة المفتاحية **case** فمثلاً لو نظرنا إلى السطر الثاني فإن الأمر أشبه ما يكون هكذا: `if ( x=='+')`

الآن لنفرض أن المستخدم قام بإدخال العددين **5** و **6** وأدخل * كعملية حسابية ، وكما تعلم فإن المتغير **x** هو العملية الحسابية وهو المتغير الذي تقوم العبارة **switch** سيبدأ تنفيذ البرنامج وسينتقل إلى السطر **2** وكما ترى فإنه لا وجود للحالة الاولى بالنسبة للعملية * ، ينتقل التنفيذ بعد ذلك إلى الحالة الثانية في السطر **5** وكما ترى فليس هناك أي مطابقة وبالتالي فسينتقل التنفيذ إلى الحالة الثالثة في السطر **8** وكما ترى فإن هناك مطابقة بالفعل وبالتالي يدخل البرنامج في هذه الحالة التي يوجد لها أمران فقط الأول يطبع القيمة والأمر الثاني يطلب من البرنامج الخروج نهائياً وترك الجملة **switch** ومواصلة سير البرنامج بشكل طبيعي وهي الكلمة المفتاحية **break** ، في حال عدم وجود الكلمة **break** فإن البرنامج سيواصل التنفيذ وسيقوم بالدخول في الحالة الرابعة وبالطبع فلا وجود لمطابقة مع المتغير **x** وبالتالي ينتقل التنفيذ إلى الحالة العامة وسيقوم بتنفيذ أوامرها بالإضافة لتنفيذه أوامر عملية الضرب ، لذلك احرص دائماً على الخروج الآمن والسليم من العبارة **switch**.

محاذير حول الجملة switch :

ينبغي لنا هنا أن نتحدث قليلاً عن القيمة التي تقوم ال **switch** بإختبارها ، تذكر أن ما تقوم هذه الجملة بإختباره هو المتغيرات وفقط ولا شيء آخر، لا تستطيع أن تقوم بكتابة أي تعبير لاختباره ، وقد ترى أن ذلك يقلل من قيمة **switch** إلا أن هذا غير صحيح فبإمكانك التوصل إلى نفس الهدف بطرق أخرى غير ما هو مفترض أو بديهي .اعتمد في هذا الأمر على تفكيرك أو حتى خيالك الواسع .

إستخدام المعاملات المنطقية مع الجملة if :

للمتغيرات المنطقية فوائد كثيرة للغاية ، إلا أنها تخفى عنا بسبب اعتمادنا الكبير على المتغيرات الأخرى وبسبب أيضاً أنها تأخرت قليلاً في الظهور في المترجمات المشهورة مثل البورلاند والفيجوال ، للمتغيرات المنطقية قيمتين فحسب واحدة منها هي صواب **true** والأخرى هي **false** ولن نقوم بوضع أي أمثلة عملية هنا بل سنترك الأمر كمهارة لك في المستقبل ، انظر لهذه الأسطر:

```
bool value=true;
if (value)
cout << "Hellow";
```

قمنا بتهيئة المتغير المنطقي **value** بالقيمة **true** ثم تقوم الجملة **if** بإختبار الشرط وهو إذا كانت قيمة **value** صحيحة أو **true** ثم تقوم بطباعتها الجملة if شبيهة بالشرط التالي:

```
if (value==true)
```

أما في حال ما أردنا العكسفبإمكاننا كتابة التالي:

```
if (!value)
```

والتي تعني السطر التالي:

```
if (value==false)
```

لاحظ هنا أننا لم نقم بوضع علامتي أقواسا لكتل الكبيرة { } والسبب في ذلك أننا لم نرد للجملة **if** أن تقوم سوى بتنفيذ جملة واحدة فحسب أما إذا أردنا كتابة أكثر من جملة فعلينا بتضمين الجمل أو الأوامر بين قوسين.

المعاملات المنطقية:

لم نناقش هذا الموضوع في الوحدة السابقة وليس السبب في ذلك عدم أهميته بل إن السبب يعود بالدرجة الأولى إلى تأجيل الموضوع لحين ظهور فائدته وبالتالي التأكيد على أهميته. تستخدم المعاملات المنطقية كثيراً في الجمل الشرطية ، والسبب في ذلك إلى أنها تناور الجملة if وتجعلها تقبل أكثر من شرط مع العلم أن الجملة **if** لا تقوم بإختبار أكثر من شرط ولكن بواسطة المعاملات المنطقية فبإمكانك جعل أكثر من شرط شرطاً واحداً وبالتالي تستطيع مناورة الجملة **if** صحيح أننا قمنا بمناقشة بعضاً من هذا الموضوع في الوحدة السابقة إلا أننا لم نتعرض لثلاث معاملات أخرى مهمة وهي:

1- معامل (و) **And** : ورمزه **&&** .
2- معامل (أو) **OR** : ورمزه || .
3- معامل (ليس) **Not** : ورمزه ! .

المعامل (And) :

هذا المعامل يقوم باختبار تعبيرين وإذا كان كلاهما صحيحاً فإنه يرجع بالقيمة **true** لنفرض أنك تقوم بكتابة برنامج يقوم باختبار درجات الطلاب وإعطاؤهم التقدير المناسب ، فإنك ستكتب لحساب التقدير ممتاز هكذا:

```
if ( (total > 90) && (total < 100 ) )
```

وبالتالي فلن تعمل الجملة **if** إلا إذا كان التعبيرين صحيحين أما إذا كان أحدهما صحيح فلن تعمل.

المعامل (OR) :

يقوم المعامل باختبار تعبيرين وفي حال كان أحد التعبيرين صحيحاً فإنه يرجع بالقيمة **true** ، لنفرض أنك تود إضافة جملة شرطية تقوم بالتأكد من أن المستخدم أدخل رقماً صحيحاً نتحدث هنا عن برنامج درجات الطلاب فإنك ستجعل الجملة if هكذا:

```
if ( (total < 0) || (total > 100 )
```

المعامل (Not) :

يقوم هذا المعامل باختبار تعبير واحد وهي تعود بالقيمة **true** إذا كان التعبير الذي يجري اختباره خطأ ، لنفرض أنك تود كتابة برنامج يقوم المستخدم من خلاله بإدخال عددين اثنين ثم يتأكد البرنامج إن كان العدد الثاني ليس قاسماً للعدد الأول ليكون قاسماً لا بد أن يكون خارج باقي القسمة يساوي الصفر، انظر لهذا الكود:

```
if ( !(numberOne% numberTwo == 0))
```

وبالتالي ففي حال كان خارج القسمة يساوي الصفر فلن يتم تنفيذ الجملة if .

مثال عملي:

سنقوم الآن بكتابة برنامج بسيط للطلاب يقوم الطالب فيه بإدخال درجته ثم يقوم الحاسب بإعطاءه التقدير ممتاز أم جيد .. إلخ. وسنستخدم في هذا المثال العبارة **else/if** والمعاملات المنطقية وبالطبع ففي نهاية هذه الوحدة سنقوم بتطوير الكود ليقدم خدمات أكثر فائدة .وربما في المستقبل تستطيع تطويره ليصبح مشروعاً رسومياً متكاملاً.

```
1- #include <iostream.h>
4-  main()
5- {
6- float degree=0;
7- char mark;
9- cout << "Please Enter Your degree:\t" ;
10- cin >> degree;
11-
12- if ((degree <=100) && (degree>= 90))
13- mark='A';
15- else if ((degree <90) && (degree>= 80))
```

```
16- mark='B';
17-
18- else if ((degree <80) && (degree>= 70))
19- mark='C';
20-
21- else if ((degree <70) && (degree>= 60))
22- mark='D';
23-
24- else if ((degree <60) || (degree>= 0))
25- mark='F';
26-
27- else if((degree >100) || (degree < 0))
28- cout << "False degree" << endl;
30- else
    {
       cout << "Bad command" << endl;
    }
32- cout << endl;
33- cout << "Your Mark:\t" << mark ;
34- cout << endl;
35-
37- }
```

الجملة:do/while

الصيغة العامة لهذه الحلقة هي كالتالي:

```
do
{
statement1;
statement2;
} while (expression) ;
```

بإمكاننا القول أن الحلقة do/while تعني قم بالدخول في الكتلة do وقم بتنفيذ الأوامر وفي حال الانتهاء قم باختبار التعبير الذي لدى الكلمة while وفي حال صحته قم بالرجوع إلى مكان الكلمة do

```
main()
{
```

```
float degree=0;
float total=0;
int i=0;
do {
cout << "Enter the degree of course number " << i+1
<< endl;
cin >> degree;
total=total+degree;
i++;
} while (i<5);
```

وبالرغم من أن استخدام هذه الحلقة قليل إلا أن ذلك لا يقلل من قيمتها وفائدتها العظيمة .وسترى أنه في بعض الأحيان لا يمكنك حل معضلة برمجية إلا بواسطة هذه الجملة.

مثال عملي:
سنقوم الآن بكتابة برنامج يقوم بكتابة جدول الضرب لأي رقم تود إظهاره ، وبالطبع سنستخدم فيه الحلقة do/while

```
4- main()
5- {
6- double number=0;
7- int i=0;
8- cout << "please Enter The Number:\t";
9- cin >> number;
10- cout << endl << endl;
11- cout << "Number\t\tOther\t\tValue"<< endl;
12- do
13- {
14- cout << number << " \t\t";
15- cout << i << " \t\t";
16- cout << i*number;
17- cout << endl;
18- i++;
19- } while ( i<=10);
21- }
```

في السطر 9 يقوم البرنامج بالطلب من المستخدم إدخال الرقم الذي يريد طباعة جدول الضرب لديه بالنسبة للسطرين 10 و 11فهي تقوم بتحسين المظهر العام للجدول. يدخل البرنامج في الحلقة do/while وتقوم الأسطر 14-17 بطباعة العددين المضروبين والناتج وتحسين المخرجات كذلك أما السطر 18 فهو يقوم بزيادة العدد الآخر المضروب زيادة واحدة حتى يستطيع البرنامج ضرب العدد الذي قمت بإدخاله في عدد آخر وتختبر الجملة while فيما إذا كان المضروب الآخر أقل من 11 وإلا فإنها ستخرج من الحلقة وبالتالي تخرج من البرنامج.

الحلقة while:

هناك فرق بين الحلقة while والحلقة do/while ففي الأخيرة يدخل البرنامج في الحلقة ثم يصطدم بالشرط أو التعبير وينتظر اختبار الشرط ، فإن كان صحيحاً أعاد التكرار مرة أخرى وإن خاطئاً استمر البرنامج في عمله دون توقف ، أما في الحلقة while فإن البرنامج يصطدم بالشرط أولاً قبل أن يدخل الحلقة ، أنظر الصيغة العامة لهذه الحلقة:

```
while (expression) {
statement1;
statement2;
statement3;
}
```

سنقوم الآن بكتابة مثال كودي بسيط حيث نطلب من المستخدم فيه كتابة ما يريد وفي حال وجد البرنامج علامة النقطة فإنه ينتهي.

```
4. main()
5. {
6. char d='a';
7. cout << "Please Enter What You want \n";
8.
9. while (d!='.')
    {
10. cin >> d;
11. }
12.
13. cout << endl << "Finish" << endl;
14.
16. }
```

انظر إلى السطر 9 ، تجد أن الشرط هو عدم إسناد المحرف (.) إلى المتغير الحرفي d وفي حال وقع ذلك فإن البرنامج يخرج من التكرار while .

ليس في المثال الحالي أي زيادة عددية ، سنقوم الآن بكتابة مثال آخر يقوم بعرض الاعداد من 0 إلى10 .

```
4. int main()
5. {
6. int number=0;
7.
```

```
8. while (number <=10) {
9. cout << "The number is :\t";
10. cout << number
11. cout << endl;
12. number++;
13. }
15. }
```

مثال عملي:

سنقوم الآن بكتابة كود يقوم بعرض الأعداد الزوجية من أي عدد يقوم المستخدم بتحديده إلى أي عدد يقوم المستخدم بتحديده أيضاً. هناك مسائل يجب أن نتناولها بعين الحذر فماذا لو قرر المستخدم أن يدخل عدداً فردياً ، لذلك علينا أن نتأكد من أن أول عدد هو عدد زوجي وفي حال لم يكن فعلينا بزيادته عدداً واحداً حتى يصبح زوجياً ، انظر لهذا المثال:

```
4.  main()
5. {
6. int number=0;
7. int max=0;
8. int min=0;
9.
10.
11. cout << "Please Enter The First Number:\t";
12. cin >> min;
13.
14. cout << "Please Enter The Last Number:\t";
15. cin >> max;
16.
17. if (!(min%2==0)) min++;
18.
19. number=min;
20.
21. while(number < max) {
22. cout << "The Next Number is\t";
23. cout << number << endl
24. number=number+2;
25. }

28. }
```

هناك ثلاثة أعداد في الأسطر 6 و 7 و 8 ، أحدهما هو أول عدد يقوم المستخدم بإدخاله حتى يبدأ البرنامج منه لعد جميع الأعداد الزوجية أما العدد الآخر فهو عدد يقوم المستخدم بإدخاله حتى ينتهي العد عنده ، أما المتغير الثالث فهو العدد الذي يستعمله البرنامج للتنقل بين الأعداد الزوجية؛ وبالطبع فإن مكمن الخطورة هنا هو أول عدد يقوم المستخدم بإدخاله فهذا العدد لو كان فردياً وابتدأ العد منه لأصبحت جميع الأعداد التي سيخرجها البرنامج أعداداً فردية. فكرة هذا المثال تقوم على التاكد من أن أول عدد هو زوجي ثم إضافة الرقم 2 إليه وطباعة العدد الجديد وهكذا حتى يصل هذا العدد إلى العدد الأخير. يقوم السطر 17 بالتأكد أن العدد المدخل الأول هو عدد زوجي وفي حال لم يكن كذلك فإنه يضيف إليه الرقم واحد حتى يصبح

زوجياً. يقوم السطر **19** بإسناد قيمة العدد الاول إلى العداد الذي سيبدأ البرنامج. العد منه وهو المتغير **number** تبدأ الحلقة **while** من السطر **21 الى السطر 25** تتم الزيادة في السطر **24** حيث يزيد العداد مرتين وليسمرة واحدة. تنتهي الحلقة **while** حينما يختل شرطها وهو أن يكون العداد أكبر من أو يساوي العدد الاكبر.

الحلقة for :

من الممكن تشبيهها بأنها عداد ينتهي عند وصول هذا العداد إلى رقم معين ثم ينتهي بعكس الحلقة **while** والتي هي تقوم بتكرير نفسها ما دام الشرط محققاً ، تأخذ الحلقة **for** الصيغة التالية:

```
for ( expr1 ; expr2 ; expr3) {
statement1;
statement2;
statement3;
}
```

حيث أن:

expr1 : هو القيمة الابتدائية للتكرار.

expr2 : وهو الشرط.

expr3 : وهو الزيادة بعد كل دورة.

سنقوم الآن بكتابة مثال يقوم بعد الأعداد من **0** إلى **10** حتى يفهم القارئ ما تعنيه الصيغة العامة للحلقة وهذا الكود هو إعادة صياغة المثال السابق.

```
4. main()
5. {
6. int number;
7.
8. for (number=0;number <=10;number++)
  {
9. cout << "The number is :\t";
10. cout << number;
11. cout << endl;
12. }
14. }
```

مثال عملي:

سنقوم الآن بكتابة مثال يقوم بجعل المستخدم يقوم بكتابة عشرة أرقام ثم يقوم البرنامج باختيار أكبر رقم وأصغر رقم ووسيلتنا لفعل ذلك هي الحلقة for بالإضافة للجملة if .

```
4. int main()
5. {
6. int number=0;
7. int max=0;
8. int min=0;
9.
10.
11. for (int i=0; i< 10;i++)
    {
12. cout << "Enter the number:\t";
13. cin >> number;
14.
15. if (number > max)
16. max=number;
17.
18. if (number < min)
19. min=number
20. }
21.
22. cout << endl << endl;
23. cout << "The Max Number is:\t" << max;
24. cout << "\nThe Min Number id:\t" << min;
25. cout << endl;
26.}
```

Ex.//write a program to calculate the SUM of positive integers and the SUM of negative integers for 6 integers.

اكتب برنامج لجمع الاعداد الموجبة والاعداد السالبة من خلال قراء ستة اعداد مختلفة ؟

```
#include <iostream.h>
main()
{
int b=0,x=0,mark=0,z=0;
for(x=1;x<=6;++x)
{
cin>>mark;
if(mark>=0)
z=mark+z;
else
b=mark+b;
}
cout<<"sum of positives="<<z<<endl;
cout<<"sum of negatives="<<b<<endl;
}
```

```
Ex.//Write a program to find N! .
#include <iostream.h>
main()
{
int a,b,c=1;
cin>>b;
for(a=1;a<=b;a++)
c=c*a;
cout<<c<<endl;
}
```

Ex.//Sum ten Numbers

```
#include <iostream>
main()
{
int number=0;
int sum =0;
for(int i=1; i<=10; i++ )
  {
     cout << "Input item number " << i << " = ";
     cin >> number;
     sum = sum + number;
   }
cout << sum;
}
```

المصفوفات والسلاسل الحرفية .

Arrays And Strings

لنفرض أنه طلب منك كتابة برنامج بسيط للغاية وهو إدخال درجات عشر طلاب ؛ لكي تحل هذا البرنامج فإن عليك أن تقوم بالإعلان عن 12 متغيراً ، وربما أن هذا مقبول نوعاً ماً ؛ ولكن ماذا لو طلب منك إدخال أكثر من درجات 1000 طالب لحل هذه الإشكالية توفر لك لغة السي بلس بلس المصفوفات.صحيح أننا قمنا بحل مسائل من هذا النوع لم تتطلب المصفوفات لكن ماذا لو طلب منك البحث عن درجة طالب معين فلن يكون هناك أي حل إلا بواسطة المصفوفات.

تعريف المصفوفات:

هي عبارة عن مجموعة من البيانات التي تشترك فيالاسم والنوع ولكنها تختلف في القيم المسندة إليها.

الإعلان عن المصفوفة:

انظر إلى هذا السطر

```
int mark[10];
```

السطر السابق هي طريقة الإعلان عن المصفوفة وكما تلاحظ فإن الإعلان يحوي ثلاثة أشياء:
نوع المصفوفة ؛ واسم المصفوفة ؛ وعدد عناصر المصفوفة.

عدد عناصر المصفوفة يجب أن يكون بين قوسين.[ ]

عدد عناصر المصفوفة اسم المصفوفة نوع المصفوفة

Int mark [20] ;

أعضاء المصفوفة:

في المصفوفة السابقة ؛ فإنها تحوي هذه العناصر:

```
int mark[0] ; int mark[1] ; int mark[2] ;
int mark[3] ; int mark[4] ; int mark[5] ;
int mark[6] ; int mark[7] ; int mark[8] ;
int mark[9] ;
```

كما تلاحظ فإن المصفوفة مكونة من عشرة عناصر حسبما هو مكتوب في الإعلان السابق . ألا ترى الرقم الملون بالأزرق هذا الرقم هو ما يسمى بدليل المصفوفة والذي يميز بين عناصر المصفوفة الواحدة ؛ المميز هنا هو أن أول عنصر فيالمصفوفة هو int mark[0] وآخر عنصر هو int mark [9] وكما تلاحظ فإنه لا وجود للعنصر العاشر وهذا ما عليك أن تعرفه وهو بالغ الأهمية العد في المصفوفة يبدأ من العنصر رقم صفر وينتهي إلى العدد ما قبل الأخير من عدد أعضاء المصفوفة المعلن عنه.

الوصول إلى عناصر المصفوفة:

حسب الشكل التوضيحي السابق فإنك تستطيع الوصول إلى أي عنصر في المصفوفة عبر كتابة نوع المصفوفة واسمها ثم دليل العنصر فمثلاً للوصول إلى أول عنصر في المصفوفة تستطيع كتابة int mark[0] وكما تلاحظ مجدداً فإن أول عنصر فيا لمصفوفة دليله هو صفر ؛ دعنا الآن من هذا الكلام النظري ودعنا ندخل لمرحلة الكتابة الكودية:

مثال عملي:

سنقوم بكتابة كود للطلاب ،عدد الطلاب فيه هو عشرة ، ثم نحسب متوسط درجات هؤلاء الطلاب ، لذلك نستطيع الإعلان عن مصفوفة مكونة من عشر عناصر وسنقوم بتسميتها int stud[10] بعد ذلك نطلب من المستخدم إدخال درجات الطلاب بواسطة دالة تكرارية ونسطيع الإعلان عم متغير من نوع int وآخر من نوع float حيث أن مهمة الأول هي حساب مجموع درجات الطلاب والثاني وظيفته قسمة المجموع على عدد الطلاب ؛ وهكذا انتهينا من حل المشكلة وبقيأن نحول الحل إلى كود وهو كالتالي:

```
1 #include <iostream.h>
2 main ( )
3 {
4 int stud[10] ,total=0 , i ;
5 float Avrege;
6 cout << "Please Enter all grades of stud:\n" ;
7 for (i=0 ; i<10 ; i++)
8 {
9 cout << "grade number" << i+1 << endl;
10 cin >> stud[i] ;
11 total=total+stud[ i ] ;
12 }
13 Avrege=total /10;
14 cout << "The Avrege of all student is: " << Avrege ;
}
```