مقدمة عامة

لغات البرمجة

تنقسم لغات البرمجة بصفة رئيسية إلى مستويين هما:

١ - لغات المستوى المنخفض Low-level languages.

٢ - لغات المستوى العالي High-Level languages.

حيث نجد أن لغات المستوى العالي أسهل في تعلمها وفهمها وذلك لأنها تستخدم كلمات إنجليزية معينة ورموز رياضية مألوفة.

١ - لغات المستوى المنخفض Low-level languages :

وتنقسم لغات هذا المستوى إلى قسمين آخرين هما:

أ - لغة الآلة Machine Language.

ب -لغة التجميع Assembly Language.

أ - لغة الآلة:

وهي اللغة الوحيدة التي يفهمها الحاسب مباشرة دون وسيط، وتعليمات هذه اللغة هي مجموعة من الأرقام الثنائية وكانت في بداية ظهور الحاسب، وكانت اللغة صعبة وخاصة أن لكل حاسب لغة آلة خاصة به، أي أنه لا يمكن نقل البرنامج من حاسب لآخر.

ولكن البرنامج المكتوب بلغة الآلة يتميز بأنه لا يحتاج إلى ترجمة.

ب -لغة التجميع أو اللغة الرمزية Assembly:

وهي مرحلة متقدمة عن لغة الآلة وأسهل نسبياً مما ساعد على إنتشار الحاسب.

وفي هذه اللغة تم استبدال الأرقام الثنائية برموز عبارة عن حرفين أو ثلاث حروف أسهل في تذكرها وكتابتها.

وتعتبر هذه اللغة مرحلة وسط بين لغة الآلة واللغات ذات المستوى العالي.

وتحتاج لغة التجميع إلى مترجم (مجمع) Assembler لترجمتها إلى لغة الآلة التي يفهمها الحاسب.

ومن عيوبها أيضاً إرتباطها بالآلة لكل آلة لها لغة تجميع خاصة بها.

برنامج المصدر(source Program) الهدف(Object Program)

٢ -لغات المستوى العالي High_Level languages :

بظهور اللغات ذات المستوى العالي أصبحت عملية التخاطب والتعامل مع الحاسب أسهل نسبياً وذلك لأن لغة التعامل مع الحاسب أصبحت قريبة من لغة البشر.

بعض مميزات هذه اللغات:

- عدم الارتباط بآلة معينة.
- سهولة تعلمها وكتابة البرامج بها.
- سهولة اكتشاف الأخطاء وتصحيحها.
- توفير الوقت والجهد.

ومن الطبيعي لبرنامج مكتوب بلغة عالية المستوى أن يترجم إلى برنامج بلغة الآلة، ويطلق على البرنامج المكتوب باللغة عالية المستوى برنامج المصدر أو كود المصدر والبرنامج المترجم يطلق عليه في هذه الحالة المترجم (Compiler).

وهنالك نوع آخر من برامج الترجمة يطلق عليه اسم المفسر Interpreter.
والمترجم أسرع من المفسر بكثير وذلك لأن المترجم يترجم برنامج المصدر مرة واحدة ثم يقوم بتخزين برنامج الهدف المكتوب بكود الآلة كملف(Object code file) وذلك لاستخدامه عند الحاجة دون أن يترجم المصدر مرة أخرى.
أما في حالة المفسر فإنه يتم ترجمة كود المصدر خطوة خطوة أثناء تنفيذ البرنامج.

مقدمة عن لغة البرمجة C++

هي لغة برمجة عالية المستوى متعددة الاستخدام، وتعتبر لغة برمجة كائنية Object Oriented Programming. يعتبرها الكثيرون اللغة الأفضل لتصميم التطبيقات ذات الواجهة الكبيرة وللتعامل مع البنية الصلبة للحاسب، وذلك لسرعتها في التنفيذ والتي لا تختلف كثيرًا عن لغة C، وفي المقابل توفر تعامل أكثر تعقيدًا مع البيانات. لغة C++ من لغات البرمجة العالية المستوى وفي نفس الوقت قريبة من لغة التجميع (بالإنجليزيةAssembly :) ذات المستوى المنخفض، كما أنها تعد لغة برمجة إجرائية (يمكن كتابة برنامج يحتوي على إجراءات وتوابع فقط) كما تعد لغة غرضية التوجه (البرنامج المكتوب عبارة عن أصناف وتستخدم الخواص المتاحة من كبسلة وتعددية الأشكال والوراثة والتركيب).

الجديد في سي++

الإضافة الأهم التي أتت بها لغة C++ عن لغة C هي البرمجة عن طريق الكائنات. حيث تعتمد لغة C على البرمجة الإجرائية والتي كانت كافية في وقتها. إلا أن ظهور أنظمة التشغيل ذات الواجهة الرسومية نقل العديد من المبرمجين إلى البرمجة بالكائنات. بالرغم من ذلك فإن لغة C ما زالت تُستخدم في برمجة الويندوز واليونكس. أبرز هذه الإضافات هي:

١- الصفوف والكائنات Classes and Objects.

٢- التحميل الزائد للعمليات: ويعتبرها الكثير أهم إضافة في C++، لأنها جعلت من C++ لغة قابلة للتوسع. هناك أكثر من ٣٠ عملية يمكن التحميل الزائد عليها.

٣- القوالب Templates.

٤- التعددية الشكليةpolymorphism

٥- الوراثة: وهي إمكانية تطوير صنف جديد يرث جميع خصائص صنف آخر. في هذه الحالة يسمى الصنف بالصنف المشتق.

٦- أستخدام الرمز // لتضمين الملاحظات بطول سطر واحد والتي يتم تجاهلها من قِبل المترجم عند القيام بعملية الترجمة.

مميزات لغة C++

بالإضافة إلى المزايا الموجودة في لغة C العديد تدعم لغة C++العديدة من المزايا الجديدة ،نذكر منها الآتي :-

١. تدعم لغة C++ البرمجة الغرضية التوجه Object Oriented Programming (OOP)
وهي تمكن المبرمج من كتابة برامج تدعم النهج الجديد في البرمجة وهو البرمجة الموجهة نحو الأشياء (OOP) والتي فيها يتم تحليل وتصميم النظام بعد تحديد مكوناته، و لكل مكون يتم تحديد خصائصه، والعمليات المعرفة عليه.

٢. شكل جديد لجمل التعليق :- command statements
حيث تم إضافة نوع آخر لجمل التعليق وهي // ويستخدم لإضافة تعليق لسطر واحد.

٣. الإعلان الحر للمتغيرات free variable declaration.
أصبح الممكن في c++ الإعلان عن المتغيرات في أي موضع من البرنامج مما يتيح ربط المتغير بالوظيفة التي من اجلها تم الإعلان عنه، مما يزيد من سهولة متابعة وفهم البرنامج .

٤. الإعلان عن الثوابت constant:-
في c++ يتم استخدام الكلمة المحجوزة const للإعلان عن الثوابت كالآتي :-

Data type const constant_ name = value
Const data type constant_ name = value

ومن مزايا هذه الطريقة تساعد المترجم على فحص الأنواع type checking وحجز ذاكرة تتناسب ونوع الثابت.

٥. المراجع:References

وهو عبارة عن اسم آخر لمتغير موجود وأكثر ما تستخدم المراجع في ++Cفي الاستدعاء بالعنوان بالدوال بدلا من استخدام المؤشرات في C .

٦. التمدد السطري Inline expansion

وهي ميزه تتعلق بالدوال ذات الحجم البسيط، حيث يتم إدخال سطور الدالة ضمن البرنامج الرئيسي أثناء زمن الترجمة حتى يتسنى تنفيذ البرنامج بشكل أسرع .

سبب صعوبتها

ربما تعود أسباب صعوبة لغتي C و++C لأسباب تاريخية أكثر منها واقعية وهو ما يردده المبرمجين المعتادين على استخدام ++C وهذا يعود بنا إلى أصل لغة C وهي لغة CPLوالتي اشتهرت بشدة تعقيدها مما تسبب في ابتعاد المبرمجين عنها فتم تطويرها وتبسيطها إلى لغة BCPL ولم تلقى الكثير من النجاح ولكنها تطورت إلى لغة البى B وعندما أرادو تطويرها سموها C(وهذا هو أصل تسمية C)، وبالطبع ورثت ++Cخصائص لغة C. إلا أنه لا يمكن إنكار أن مفاهيم مثل المؤشراتpointers ، والإشارات هي مفاهيم مربكة للمبرمج المبتدئ، بالإضافة إلى مفاهيم التوارث Inheritance وتعدد الشكل polymorphism والقوالب Templates التي تربك حتى المبرمجين المحترفين.

أساسيات لغة ++C:

ربما أفضل طريقة لتعلم لغة برمجة هي كتابة برنامج لذلك لدينا البرنامج التالي:

```
1  // my first program in C++
2  #include <iostream>
3  using namespace std;
4  int main ()
5  {
6    cout << "Hello World!";
7    return 0;
8  }
9
10
```

```
Hello World!
```

عودة لشكل البرنامج بلغة++C

لاحظ البرنامج كالتالي:

```
1 #include <iostream.h>
2 // This program print " Welcome to C++" on the screen
3 using namespace std;
4 int main()
5 {
6 cout << "Welcome to C++\n";
7 return 0;
8 }
```

المخرج من البرنامج بعد التنفيذ:

```
Welcome to C++
```

شرح أسطر البرنامج

السطر الأول:

```
#include <iostream>
```

كلمة include عبارة عن توجيه للمترجم بأن يضم محتويات ملف الرأس iostream إلى الملف الحالي بحيث يصبح جزءاً منه. ويوجد ملف الرأسiostream (وهو اختصار input/output stream في مكتبة C++ القياسية)، ويحتوي على التعاريف و الإعلانات الضرورية والمفيدة لعمليات الإدخال و الإخراج القياسية. حيث تتعامل هذه المكتبة مع عمليات الإدخال و الإخراج على أنها تيارات من الحروف.

يجب إضافة ملف الرأس iostream في أي برنامج يستخدم أدوات الإدخال أو الإخراج القياسية مثل cin و cout. ، والرمز # الذي يسبق include يستخدم للدلالة على أن السطر الحالي هو توجيها للمترجم. أما قوسا الزاوية < > اللذان يحيطان بالملف iostream فيدلان على وجود ملف الرأس هذا في مكتبة C++ القاسية ثم يقوم المعالج المبدئي بتنفيذ مثل هذه التوجيهات والتي تبدأ بـ#.

السطر الثاني :

```
// This program print " Welcome to C++" on the screen
```

يعتبر هذا تعليقاً (comment) خاص بالمبرمج، وهو عبارة عن نص يضاف في الملف المصدر من أجل وصف أو شرح عمل البرنامج وتوثيقه ، وهذا التعليق ليس اجباريا في البرنامج ولكن ينصح بإضافتها إلى نصوص البرامج.
ويمكن كتابتها في لغة C++ بطريقتين :

الطريقة الاولي : تعليق أحادي السطر: ويبدأ بكتابة الخطين المائلين // في السطر وينتهي بنهاية نفس السطر. وسيعتبر المترجم أن أي شي يبدأ من الخطين المائلين إلى نهاية السطر تعليقاً.

الطريقة الثانية : تعليق متعدد السطور: وهو النمط القديم المستخدم في لغة C. ويبدأ بـ /* (خط مائل تليه نجمة) وينتهي بـ */(نجمة يليها الخط المائل)، مثل/* Welcome to my first program in C++ */ :
ويمكن استخدامها عند كتابة التعليقات الطويلة ولكن لا يسمح بتداخلها مع بعضها البعض، وسيتجاهلها المترجم كذلك.

السطر الثالث :

```
using namespace std;
```

تخبر المترجم بأن يستخدم فضاء الأسماء المعرف باسم std اختصارا لكلمة standard. وفضاء الأسماء (namespace)ميزة جديدة أضيفت إلى C++ بغرض حل مشكلة تضارب الأسماء التعريفية خاصة عند استخدام

السطر الرابع :

```
int main()
```

كل برنامج من برامج الـ C++ يحتوي على إجراء واحد أو أكثر، ولكل منها أسم واحد فقط من هذه الإجراءات يجب أن يحمل الاسم main() ، وهو الإجراء الرئيسي الذي يبدأ منه تنفيذ البرنامج و غالبا ما ينهي منه.
أما القوسين المجودة بعد كلمة main فهما ضرورية، على الرغم من عدم وجود شي بينهما، للدلالة على أنه إجراء.
وتستخدم هذه الأقواس التي تلي أسماء الإجراءات من أجل تمرير المتغيرات الوسيطة .

أما القوسان الكبيران { } في السطرين ٥ و ٨ ضروريان لتحديد بداية ونهاية الأوامر في الإجراء. أي من أجل تحديد محتوى الإجراء.main()

السطر الخامس :

```
cout << " Welcome to C++ \n";
```

هذه عبارة cout التي تقوم بعرض الرسالة "Welcome to C++" على الشاشة. ولغة C++ تنجز عمليات الإدخال والإخراج على شكل تيارات من الحروف .(streams of characters)

تسمى الرسالة "Welcome to C++ \n" والمحصورة بين علامتي الاقتباس المزدوجة بسلسلة حروف. وعند التنفيذ، تطبع السلسلة بالصورة التي تبدو فيها للعيان وهي محصورة بين علامتي الاقتباس، باستثناء الحرفين الأخيرين n\يستخدم الخط المائل (\) كرمز للهروب، وعندما يظهر في سلسلة حروف، فإن الحرف الذي يليه ينضم معه ليشكل تتابع من الحروف تستخدم لترميز حروف خاصة. فالتتابع n\ يرمز لسطر جديد. وتؤدي إلى تحرك مؤشر الشاشة إلى بداية السطر التالي.

السطر السادس :

```
return 0;
```

العبارة return تودي إلى إنهاء الإجراء وإعادة القيمة صفر إلى الإجراء المستدعي (نظام التشغيل). ومن المتفق علية أن يعيد البرنامج القيمة صفر عندما يتم تنفيذه بالصورة الطبيعية دون حدوث أي مشاكل أو أخطاء. أما إذا حدث خطأ فإن البرنامج يعيد قيمة أخرى غير الصفر يستدل بها في تحديد نوع الخطأ.

الخلاصة:
١- الأسطر التي تبدأ بالعلامة # تعتبر توجيهات للمعالج المبدئي preprocessor directives مثل الأمر #include<iostream>.ونظراً لأهمية ملف الرأس iostream في عمليات الإدخال والإخراج وكذلك كون الفضاء std هو فضاء مكتبة C++ فإن كل برنامج سيحتوي على السطرين:

```
#include <iostream>
using namespace std;
```

٢- يحتوي كل برنامج C++ على إجراء رئيسي يسمى main() والذي منه يبدأ تنفيذ البرنامج، و تعليمات أي إجراء يجب أن تحصر بين الأقواس الكبيرة.{ }

٣- في الوضع الافتراضي، يمثل الكائن cout الشاشة، وعند إدراج أي شي فيه (سلسلة من الحروف) يتم عرضه على الشاشة. ويوجد تعريفه في مكتبة النظامiostream ، ويمكن لأي إجراء آخر استخدامه.

٤- وضع التعليقات مهم لقرّاء نص البرنامج، أما المترجم فيتجاهلها تماما.

٥- تنتهي كل تعليمة بفاصلة منقوطة (؛)وتعبّر عن نهاية التعليمة.

المراحل التي يمر بها البرنامج بلغة ++C قبل تنفيذه:

١.مرحلة كتابة البرنامج. Edit
كتابة البرنامج ،وفيها تتم كتابة البرنامج بواسطة محرر النصوص الخاص بالمترجم والذي سيقوم بالتدقيقات والبحث عن الأخطاء اللغوية ، أما الأخطاء المنطقية فهي تتم من خلال تجريب البرنامج والبحث عن المخرجات والتمعن فيما إذا كانت تماثل المخرجات العقلية أم لا.

٢.مرحلة ما قبل الترجمة او المعالجةPreprocess

عندها يتم التعامل مع مجموعة من التوجيهات والإجراءات الواجب تنفيذها على نص البرنامج قبل عملية الترجمة ، اي (معالجة البرنامج قبل عملية الترجمة) وتشبه هذه العملية المعالجة الأولية لمحرر النصوص وتعمل هذه العملية على الإضافات اللازمة والتعديلات اللازمة على البرنامج المصدري.

٣.مرحلة الترجمةCompile
أي تحويل البرنامج المكتوب بلغة الC++ إلى لغة الآلة . أي تحويل الكود من لغة ++C إلى لغة الآلة ، ولكن هذا الكود غير قابل للتنفيذ حتى الآن.

٤.مرحلة الربطLink
وهي ربط النصوص مع بعضها بالطرق المنطقية ، أي : أن برامج ++C تحتوي على استدعاء دوال تم تعريفها في أماكن مختلفة مثل : المكتبات المعيارية ، او المكتبات الخاصة بمجموعة من المبرمجين لتسهيل العمل عليهم مستقبلاً .
يحتوي الملف المراد ترجمته على نقص في عملية الترجمة فيقوم الرابط بعملية الربط بين النقص الحاصل في الملف المراد ترجمته وأماكن تواجد النصوص الناقصة ، وهذا يعني الربط بين استدعاء الدالة ومكان تعريفها.
ينشأ لنا بعد ذلك ملف تنفيذي يمثل الصورة الأساسية للبرنامج ، بعدها تأتي عملية التحميل.

٥.مرحلة التحميلLoad
ونقصد بهذا المفهوم تحميل البرنامج إلى الذاكرة لتنفيذه ويقوم بهذه العملية المحمل الذي ينقل الملف التنفيذي إلى الذاكرة.

٦.مرحلة التنفيذExecute
وبعدها يبدأ الحاسب بتنفيذ البرنامج بشكل متسلسل حسب التعليمات ، أي تعليمة تلو الأخرى حسب إشراف وحدة المعالجةCPU

أنواع البيانات في ++C:-

هناك أنواع أساسيه وأنواع مشتقة من الأنواع الأساسية ويمكن تلخيصها في الجدول التالي :-

Type	Length	Range
Unsigned char	8 bits	0 to 255
char	8 bits	-128 to 127
enum	16 bits	-32,768 to 32,767
unsigned int	16 bits	0 to 65,535
short int	16 bits	-32,768 to 32,767
int	16 bits	-32,768 to 32,767
unsigned long	32 bits	0 to 4,294,967,295
long	32 bits	-2,147,483,648 to 2,147,483,647
float	32 bits	3.4 * (10**-38) to 3.4 *(10**+38)
double	64 bits	1.7 * (10**-308) to 1.7 * (10**+308)
long double	80 bits	3.4 * (10**-4932) to 1.1 * 10**+4932)

ملحوظة :-

١. الأنواع المشتقة: **هي كلمتي long، short وهي تستخدم مع أي من الأنواع الأساسية مثل :-**
long int =4bytes short int =2bytes
٢. إذا كتبنا shortأوlong فقط بدون إلحاقها بأي من الأنواع الأساسية فإن المترجم يأخذها كما يلي:- = short int
٣. جميع الأحرف تكتب بالأحرف الصغيرة (small letters).
٤. يمكننا في ++c إضافة أنواع جديدة من البيانات مثل:-
enum colors {red , blue , green}

enum cars {Toyota , BMW ,Mitsubishi}

المعرفات:-

هي عبارة عن أسماء تستخدم في البرنامج لغرض معين أو وظيفة معينة. وتمثل المتغيرات بكافة أنواعها والثوابت و أسماء الدوال.

شروط تسمية المعرفات:-

١. يبدأ بحرف، وتليه أي مجموعة من الحروف أو الأرقام.
٢. لا يحوي فراغات، ولكن يمكن أن يحوي(_)Under Line Character.
٣. لا يحوي رموز رياضية {%, +, - , / , *}،
ولا الأحرف الخاصة { /,\,space; ,?,",:, ', !} .
٤. لا يكن احدي الكلمات المحجوزة في C++.
٥. أن يعكس الوظيفة التي يؤديها.
٦. تفرق لغة C++ بين الحروف الكبيرة والصغيرة .

المتغيرات:-Variables

المتغير هو عبارة عن مساحة من الذاكرة محجوزة باسم معين، وتستخدم لحفظ واسترجاع البيانات أثناء تنفيذ البرنامج.

ملحوظة :- تخضع تسمية المتغيرات لنفس شروط المعرفات.

الإعلان عن المتغيرات:-

قبل استخدام أي متغير في C++، يجب أولاً الإعلان عنه وذلك بتحديد نوعه واسمه.

الصيغة العامة للإعلان عن المتغيرات:-

```
Data type Variable name ;
```

أمثلة لبعض الإعلانات للمتغيرات:-

```
int a;
float incom;
double x,y,z;
double weight ;
 char ch;
```

إعطاء المتغير قيمة ابتدائية:-initial value

في بعض الأحيان نحتاج قيمة استهلالية للمتغير، ويتم إعطاء هذه القيمة للمتغير عند الإعلان عنه، مثلاً:

```
int sum = 0;
float salary= 0.0;
char ch=' ';
```

الثوابت:-Constant

هي عبارة عن متغيرات تحمل قيم ثابتة طيلة زمن تنفيذ البرنامج، ويجب أن تأخذ الثوابت قيمها عند الإعلان عنها مباشرة، ولا يمكن تغيير قيمتها بعد ذلك. وتتبع الثوابت نفس شروط تسمية المتغيرات. ويتم الإعلان عن الثوابت باستخدام الكلمة const، قبل كتابة النوع أو بعده.

الصيغة العامة للإعلان عن الثوابت :

```
Data type const constant_name = value
Const data type constant_name = value
```

أمثلة:

```
float const pi = 3.14;
int const max =50;
```

أو يمكن كتابتها كالآتي:-

```
const float pi =3.14;
const int max =50;
```

أوامر الإدخال والإخراج للبيانات:

جملة الإدخال الأساسية (أمر قراءة البيانات) **cin>>**.

لإدخال قيمة للمتغير **cin>>variable name;**

مثلاً:

cin>>a;

ملاحظة: يمكن أن تحوي جملة الإدخال أكثر من متغير على أن تفصل بينها إشارة >>

مثال:

cin>>a>>b>>c

جملة الإخراج الأساسية :أمر الطباعة (إخراج البيانات على الشاشة)cout<<.
cout<<list لإخراج سلسلة على الشاشة

حيث list يمثل أسماء متغيرات أو أسماء ثوابت

أو تعبير منطقي أو حسابي يراد إخراج نتيجة معالجته مباشرة على شاشة الحاسوب

أو تعليق على أن يوضع ضمن إشارتي اقتباس أو أحد رموز التنسيق التي سنتعرف عليها لاحقاً

أمثلة:

cout<<x;

cout<<x+y<<5*z;

cout<<x<<y<<"the results="<<z;

أحرف الهروب المتتالية :-Escape Sequence Characters

هي عبارة عن أحرف تستخدم في عبارة cout و يكون ناتجها عبارة عن حركة في موضوع المؤشر و لا يتم طباعتها.
فمثلاً، إذا أردنا طباعة سطر جديد، فإننا نستخدم أحد أحرف الهروب وهو '\n'، كالآتي:-

Cout<<"Hello,\nWorld"	**Output:-** **Hello,** **World**

الجدول التالي يوضح كل أحرف الهروب في C++، ووظيفة كل منها:

الحرف	الوظيـــــــــفة
\n	تطبع سطر جديد
\t	تطبع tab،(أربع مسافات)
\a	صوت(Alert)
\f	تقوم بنقل المؤشر إلى السطر التالي. (Page Feed)

\?	تطبع علامة استفهام (?)
\b	(Back Space)ترجع المؤشر خطوة للخلف.
\\	تطبع \
\r	تنقل المؤشر إلي بداية السطر الحالي (Carnage Return)
\v	تطبع مسافة رأسية |
\'	تطبع علامة تنصيص'
\"	تطبع علامتي تنصيص"

العمليات الحسابية:(+, -, *, /, %) Arithmetic operators

أولاً: العوامل (المؤثرات) الرياضية Arithmetic Operators :

هناك خمس مؤثرات حسابية في لغة ++c وهي :

المؤثر (العامل) operator	معناه
+	الجمع
-	الطرح
*	الضرب
/	القسمة
%	باقي القسمة

ملاحظات:

عند تنفيذ أي تعابير رياضية يستخدم فيها المؤثرات الحسابية السابقة يجب أن تنفذ بالترتيب التالي:

١ -الأقواس.( )

٢ -الأس. ^

٢- الضرب والقسمة وباقي القسمة من اليسار إلى اليمين.(%,/,*)

٣- الجمع والطرح من اليسار إلى اليمين.(-,+)

كما يجب أن يؤخذ في عين الاعتبار في حالة استخدام مؤثر القسمة ان الناتج سيكون عدد صحيح في حالة أن المتغير المستخدم لتخزين ناتج القسمة تم تعريفه على أنه عدد صحيح.

مثال: أكتب برنامج بلغة C++ يطلب من المستخدم إدخال عددين ثم يطبع ناتج جمع العددين.

```
#include <iostream.h>
#include <conio.h>
 int main()
}
  int x,y,sum;
  cout<<"enter number1";
  cin>>x;
  cout<<"enter number2";
  cin>>y;
  sum=x+y;
```

```
  cout<<sum;
 getch();
{
```

ملحوظة :

نلاحظ أننا عند نهاية الدالة الرئيسية (()main)، قمنا بكتابة الأمر()getch، وهي عبارة عن دالة معرفة مسبقاً في المكتبة (conio.h)، وهذه الدالة تطلب من المترجم قراءة حرف من لوحة المفاتيح، وقد تم استخدامها هنا ليس لغرض قراءة الحرف، وإنما لتثبيت الشاشة حتى نتمكن من رؤية الناتج علي الشاشة.

مواصلة لأنواع المؤثرات في لغة C++

أولاً: العوامل (المؤثرات) الرياضية Arithmetic Operators (تمت مناقشته في المحاضرة السابقة)

ثانياً: مؤثرات المقارنة Relational Operators :

توجد ست مؤثرات مقارنة يستطيع المبرمج استخدامها لمقارنة قيمتين وينتج عن عمليات المقارنة قيمتين فقط وهما:

False أي قيمة خاطئة

True أي قيمة صحيحة

ويوضح الجدول التالي مؤثرات المقارنة في لغة C++:

المؤثر operator	معناه	مثال
==	يساوي	(x==y)
=!	لايساوي	(x!=y)
<	أقل من	(x<y)
<=	أقل من أو يساوي	(x<=y)
>	أكبر من	(x>y)
>=	أكبر من أو يساوي	(x>=y)

ثالثاً: المؤثرات المنطقية logical Operators :

هناك عدد من المؤثرات المنطقية التي تنتج عنها قيمتين إما قيمة صحيحة true أو قيمة خاطئة false وهذه المؤثرات هي:

المؤثر operators	معناه
&&	AND هو مؤثر ثنائي تكون قيمته صحيحة عندما يكون كلا التعبيرين صحيحين

||	OR مؤثر ثنائي تكون قيمته صحيحة عندما يكون أحد التعبيرين صحيحا
!	مؤثر أحادي NOT نفي التعبير أو النقيض

رابعاً: مؤثرات الزيادة والنقصان Increment Decrement Operators:

تتميز لغة C++ بمؤثر الزيادة ++ ومؤثر النقصان – حيث يمكن استخدامها مع المتغيرات فقط وتستخدم بالشكل التالي :

op variableName;

وتكافئ

VariableName op;

حيث op مؤثر الزيادة أو النقصان

VariableName اسم المتغير الذي تم الإعلان عنه مسبقا.

يمكن وضع مؤثر الزيادة ++أو النقصان -- في بداية المتغير أو في نهايته ولكن يجب الأخذ في الاعتبار أن هناك فرق بين الموضعين المثال التالي يوضح الفرق.

مثال١:

```
int i=2,j=2;
int k=++i;
k=j++;
```

في الجملة الأولى مؤثر الزيادة ظهر في بداية المتغير i وهنا سيضاف العدد 1 على قيمة i أولا ثم بعد ذلك تخزن النتيجة النهائية في المتغير k اي ستكون قيمة المتغير k تساوي 3

أما في الجملة الثالثة مؤثر الزيادة ظهر في نهاية المتغير فأولا سيعين قيمة المتغير k وهي تساوي قيمة المتغير j اي أن k يساوي 2 ثم بعد ذلك يزيد قيمة المتغير j بالعدد 1 حيث تصبح قيمة المتغير j تساوي 3

خامساً: المؤثرات المركبة Compound Operators:

يمكن استخدام المؤثرات الحسابية (%,*,/,-,+)مع مؤثر التخصيص (=) تحت اسم يعرف بالمثرات المركبة وهي تعتبر طريقة مختصرة لجملة التخصيص أو التعيين.

الشكل العام: VariableName op = Expretion;

حيث op تمثل احدى المؤثرات الحسابية .

مثال:

x+=9; تكافئ x=x+9;

وكلا الجملتين (التعبيرين) معناهما أضف العدد 9 للمتغير القديم x ثم خصص هذه القيمة للمتغير الجديد x الموجود في الطرف الأيسر.

المؤثر المركب	مثال
+=	a+=6 وتعني a= a+6
-=	a-=5 وتعني a=a-5
*=	a*=5 وتعني a=a*5
/=	a/=3 وتعني a=a/3
%=	a%=7 وتعني a=a%7

الأخطاء الشائعة في لغة C++

من المعروف في عالم البرمجة أن المبرمج يدخل في أخطاء غير متعمدة في البرنامج الذي يعده وبالفحص الدقيق للأسباب العامة للأخطاء يستطيع المبرمج ان يتلافى عدد كبيراً منها، وهذه العملية تسمى Debugging أي البحث عن الأخطاء وإصلاحها وفي هذه الفقرة نقدم بعض الإرشادات المهمة التي يمكن من خلالها اكتشاف هذه الأخطاء وإصلاحها والتغلب عليها وهذه الأخطاء يطبعها الحاسب في نهاية البرنامج كقائمة أخطاء، هذا و يمكن أن تصنف الأخطاء التي تقع عموماً إلى الأنواع التالية:-

١ - **أخطاء ناتجة عن ترجمة البرنامج إلى لغة الآلةCompilation Errors:-**

- ✓ مخالفة قواعد التنقيط من فواصل وأقواس ونقاط.
- ✓ الأخطاء في كتابة وإملاء الكلمات.
- ✓ استخدام رموز أو أحرف غير مقبولة.

٢- **أخطاء تنفيذية (أثناء تشغيل البرنامج) Execution Errors:-**

- ✓ إذا كان هنالك خطأ في تعريف أحد المتغيرات الحسابية أو المنطقية أو الرمزية.
- ✓ إذا كانت نتيجة الحساب فوق طاقة الحاسب مثلاً 10^8.

٣- أخطاء منطقية Logical Errors:-

- ✓ وجود عملية تكرار خاطئة في البرنامج.
- ✓ وجود خطوات لاتخاذ قرارات خاطئة في البرنامج.

مثال: أكتب برنامج يطلب من المستخدم إدخال ثلاثة أعداد ثم يطبع مجموع وطرح وضرب وقسمة ومتوسط الأعداد.

مثال: اكتب برنامج يقوم بحساب مساحة و محيط الدائرة من المعادلتين

$A= \pi R^2$

$H=2 \pi R$

تمارين

استخدام أحرف الهروب:

١- أكتب برنامج يقوم بطباعة حاصل جمع و طرح و ضرب وقسمة عددين صحيحين وأن تظهر النتيجة بالشكل التالي ؟

```
#include <iostream.h>
#include <conio.h>
void main()
{
   int x , y;
   cout<<"Enter First number :-";
   cin>>x;
   cout<<"Enter Second  number :-";
   cin>>y;
   cout<<x<<"\t+\t"<<y<<"\t = \t"<<  x+y <<"\n";
   cout<<x<<"\t-\t"<<y<<"\t = \t"<<  x-y <<"\n";
   cout<<x<<"\t*\t"<<y<<"\t = \t"<<  x*y <<"\n";
   cout<<x<<"\t/\t"<<y<<"\t = \t"<<  x/y <<"\n";
   getch( );
}
```

المخرج

```
Enter First number : 10
Enter Second  number : 5
10   +   5   =   15
10   -   5   =   5
10   *   5   =   50
10   /   5   =   2
```

مؤثرات الزيادة والنقصان:

مثال: تحقق من ناتج البرنامج التالي:

```
#include <cstdlib>
#include <iostream>
using namespace std;

int main()
{    int x=10;
cout<<x++<<"\n";
cout<<x<<"\n";
cout<<++x<<"\n";
cout<<x<<"\n";
cout<<x--<<"\n";
cout<<x<<"\n";
cout<<--x<<"\n";
cout<<x<<"\n";
     system("PAUSE");
     return EXIT_SUCCESS;
}
```

مثال:-

أكتب برنامج يقرأ الراتب الأساسي لموظف ثم يقوم بحساب الأتي :-

١. المكافأة =٧% من الراتب الأساسي .
٢. المستقطع= ٥% الراتب الأساسي.
٣. صافي الراتب.

جمل التحكم(الجمل الشرطية) Control Statements

يقصد بالجمل الشرطية ببساطة هي الجمل التي قد تنفذ أو لا ، حسب قاعدة نحن نكتبها أو الأصح شرط نحن نكتبه , وسميت تحديديه Selection لأنها نحدد ما الذي ينفذ من الجمل:

وتنقسم جمل التحكم إلي قسمين:

١) **جمل التفريع: Branching Statements**

وفيها يتم تنفيذ جملة أو عدة جمل وتجاهل أخري بناءاً علي شرط معين. وتحوي جملتين هما:-

- **جملة If**
- **جملة Switch**

٢) **جمل التكرار:- Iterative Statements**

وفيها يتم تكرار تنفيذ جملة أو مجموعة من الجمل لعدد معين من المرات أو إلي حين تحقق شرط معين. وتنقسم إلي قسمين:-

- **جمل التكرار الثابت:-Fixed Loops**
  - **حلقة for**
- **جمل التكرار المتغير:-Variables Loops**
  - **حلقة While**
  - **حلقة Do..While**

أولاً: جمل التفريغ:-Branching Statements

الجملة الشرطية If :

تستخدم لتنفيذ جملة (أو عدة جمل) أو تجاهلها بناء على شرط معين. ولها ثلاث صور:-

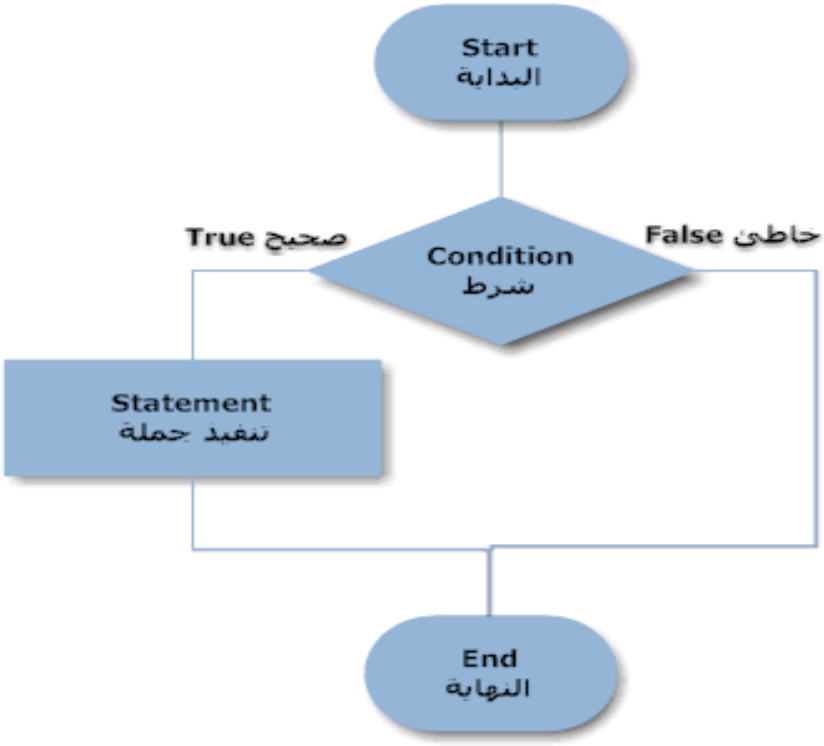
١. **IF البسيطة:-Simple If**

الجملة if هكذا تكتب ولا تكتب IF او iF او If كما تعلمنا أن لغة++C حساسة لحاله الأحرف، هي جملة في ابسط أشكالها تختبر شرط ما إذا كان الشرط صحيح فإنه ينفذ الجملة التابعة لجملة if وإذا كانت خاطئة فيتجاوز ما يتبع جملة if ويتابع عمله...

والصيغة العامة تكون كالتالي:**كود:**

```
if (condition )
{
   statement1;
   statement2;
   statement3;
.
.
.
}
```

في هذه الصيغة نلاحظ أن هنالك قوسين { و } وهي عبارة عن بداية محتوى جمل if ونهايتها، إذن في حال تنفيذ الجملة فإذا تحقق الشرط فإنه يتابع لينفذ الجملة او البلوك التابع لجملة if وإذا لم يتحقق فإنه يتجاوز مباشرة عن هذه الجملة .... لاحظو الشكل التالي:



وهذه جملة الـ if في ابسط حالاتها ، ونلاحظ أنها لم تنتهي بالفاصلة المنقوطة ، وإذا وضعت الفاصلة المنقوطة فإن هذا يعتبر إنهاء لجملة if وتعتبر بنية كاملة ، اعتقد بالأمثلة تصل الفكرة التي قد لا تصل حقيقة إلا بالأمثلة.. **كود:**

```
int main( )
{  int age;
  cout<<"Please enter your age: ";
  cin>> age;
  if ( age < 60 ) {
     cout<<"You are pretty young!\n";
  }
}
```

أنظر للبرنامج هو يقوم بقراءة رقم نحن أدخلناه خلال تنفيذ البرنامج ، يقرأ البرنامج قيمة المتغير ageبعد الإدخال ومن ثم يرى هل هو اكبر من القيمة ٦٠ فإذا كان ذلك، يطبق الجملة التي داخل البلوك التابع له فإذا كان العمر المدخل اقل من ٦٠ يطبع !You are pretty young أما إذا يساوي ٦٠ أو اكبر من ٦٠ فإنه لا يطبع شيء .

٢. جملة : if / else

هذه الجملة تختلف عن السابقة باختلاف بسيط وهي انه إذا لم يتحقق شرط فهنالك بديل أي إذا لم يتحقق الشرط فقم بشئ آخر.

الصيغة العامة لـ: if /else

كود:

```
if (condition )
```

```
{
  statement1;
   statement2;
   statement3;
.
.
.
}
else
{
  other statement1;
other statement2;
other statement3;
.
.
.
}
```

مثال على ذلك:

هنا نريد أن نضيف شيئا للمثال السابق وهو ببساطة إذا لم يتحقق الشرط يطبع له انه كبير في السن ، فهذا يتحقق عن طريق else التي تتبع if .

كود:

```
int main()
{
 int age;
 cout<<"Please input your age: ";
 cin>> age;
 if ( age < 60 )
{
   cout<<"You are pretty young!\n";
 }
else
 {
  cout<<"You are old\n";
 }

}
```

تمرين: أكتب برنامج يطلب من المستخدم إدخال عددين ثم يقارنهما ويطبع العدد الأكبر

٣. جملة : if /else if

هذا الشكل بسيط فهمه اذا فهمنا النقطة السابقة باختصار هي اذا تحقق شرط ينفذ جملة ، اذا لم يتحقق يقوم بتنفيذ جملة if ثانية لكن فقط في حال عدم تحقق الشرط الاول ، كما في الصيغة التالية:

```
If (condition1)
    Statement(s);
Else If(condition2)
        Statement(s);
    Else If(condition3)
            Statement(s);
        Else
            Statement(s);
```

حسب هذه الصيغة بإمكاننا تطبيق اي نوع من انواع if او if/else بكل وحدة ، لفهم هذا في المثال التالي:

مثال :- أكتب برنامج يطلب من المستخدم إدخال درجة طالب(mark) ويطبع تقدير الطالب وذلك وفق الآتي

إذا كان mark>89 يطبع Excelent
إذا كان mark>79 يطبع Very Good
إذا كان mark>69 يطبع Good
إذا كان mark>59 يطبع Pass
وإلا يطبع Fall

```
int main ( )
{  int mark;
cout<<"Enter Mark";
  cin>>mark;
     if ( mark > 90 )
     cout<<"EXCELLENT"<<endl;
  else if ( mark > 80)
    cout<<"VERY GOOD"<<endl;
  else if ( mark > 70)
    cout<<"GOOD"<<endl;
else if ( mark > 60 )
    cout<<"PASS"<<endl;
else {
    cout<<"FALL"<<endl;
    cout<<"TRY ANOTHER TIME"<<endl;
{
cout<<"thank                                                    you"<<endl;
}
```

٢.جملة switch

جملة switch هي جملة شرطيه مثلها مثل if ولكن تختلف عنها في أن switch تقوم باختبار قيمة (متغير) واحدة وليس شرط مثل ما تفعل if .. بكلام آخر فإن switch تقوم بتنفيذ كود معين بناءاً على قيمة معينة وللتوضيح :

الصورة العامة :-

```
switch( Variable)
{
case Value1 : Statement (s); break ;
case Value2 : Statement (s); break ;
case Value3 : Statement (s); break ;
.
.
case Value n : Statement (s); break ;
default : Statement (s);
}
```

مثال: برنامج يقوم بطباعة اليوم حسب العدد المدخل.

```
int day;
cout<<"enter the day number\n";
cin>>day;
switch (day)
{
case 1 :cout<<" The day is Saturday\n" ; break;
case 2 :cout<<" The day is Sunday\n" ; break;
case 3 :cout<<" The day is Monday\n" ; break;
case 4 :cout<<" The day is Tuesday\n" ; break;
case 5 :cout<<" The day is Wednesday\n" ; break;
case 6 :cout<<" The day is Thursday\n" ; break;
case 7 :cout<<" The day is Friday\n" ; break;
default :cout<<" Wrong day number\n";
}
```

الآن ببساطة الذي تعمله switch هو اختبار لقيمة المتغير day ومعنى كلمة case أي أنه في حالة كذا نفذ الجزء الخاص بي حتى عبارة break . بكلام آخر فإن switch تقوم باختبار المتغير عند أكثر من قيمة وفي حالة مطابقة القيمة فإن الـ case بتلك القيمة يتم تنفذ الكود الخاص بها.

أما بالنسبة لعبارة default فهي تمثل الجزء الذي يتم تنفيذه في حالة لم يتم تنفيذ أي case أي لا يوجد مطابقة بين قيمة المتغير وبين جميع الـ case الموجودة عنها يتم تنفذ default .

ملاحظة : يجب الانتباه إلى وضع جملة break لأن وظيفتها إظهار النتيجة بعد تحقق الشرط مباشرة

وبالتالي يقل زمن تنفيذ البرنامج وفي حال تم نسيان الجملة فإنها ستقوم بالدخول على الـ case التي تليها مباشرة

شروط استخدام switch. case:-

١. يجب أن يكون متغير switch من النوع الصحيح أو الحرفي.
٢. يجب أن لا يكون متغير switch عبارة عن تعبير رياضي أو منطقي.
٣. يجب أن يكون متغير switch، و قيم الخيارات من نفس النوع.
٤. يجب أن تحوي الخيارات قيم ثابتة وليست متغيرة.

مثال:-

اكتب برنامج يقرأ رمز رياضي و رقمين، وحسب الرمز الرياضي المدخل يقوم البرنامج بإجراء العملية المناسبة علي الرقمين وطباعة الناتج.

الحل:

```
int main( )
{   int x, y;
    char sign;
    cout<<" Enter Two Numbers \n";
    cin>>x;
    cin>>y;
    cout<<" Enter The sign\t ";
    cin>>sign;
    switch( sign )
    {
  case '+':cout<<x<<" + "<<y<<" = "<<x+y<<"\n";break;
  case '-':cout<<x<<" - "<<y<<" = "<<x-y<<"\n";break;
  case '*':cout<<x<<" * "<<y<<" = "<<x*y<<"\n";break;
  case '/':cout<<x<<" / "<<y<<" = "<<x/y<<"\n";break;
  default : cout<<"wrong sign try again...";
    }
}
```

```
Output=
Please Enter
Two
Numbers
2
5
Please Enter
The sign
+
2+5=7
```

حلقة For :

تقوم بتكرار جملة أو عدة جمل إلى عدد معين من المرات .او الى حين يتحقق شرط معين.

الصيغة العامة :-

```
For(Initial-Value; Condition; Increment)

    Statement(s);
```

حيث:-

Initial-Value هي القيمة الابتدائية للحلقة.
Condition هو شرط التوقف من الحلقة.
Increment هو معيار الزيادة أو التناقص(خطوة الحلقة).

مثال:-

لطباعة الجملة "Hello World" عشر مرات علي الشاشة، فإننا نكتب:-

```
for(int i=0;i<=10;i++)
```

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

في هذا المثال، i هو المتغير العداد للحلقة الذي يتابع عدد مرات تنفيذ الحلقة، وقد أخذ القيمة الابتدائية 0. في C++، يمكن الإعلان عن المتغيرات في أي موضع من البرنامج كما علمنا مسبقا وهنا يمكن دمج الإعلان عن المتغير العداد مع الحلقةforً. شرط التوقف من الحلقة هو عندما تصل قيمة العداد i=10، والذي يتزايد بمقدار 1 في كل دورة.

يمكن كتابة نفس المثال السابق بحلقة تناقصية كالآتي:-

```
for(int i=10;i>0;i--)
    cout<<"Hello World";
```

في هذه الحالة، فإن قيمة المتغير العداد تتناقص بقيمة واحدة في كل دورة.

أمثلة:-

مثال(١):-

اكتب برنامج يقوم بطباعة الاعداد من ١ إلى ١٠٠؟

لو استخدم المخرجات لتطلب الامر ١٠٠ سطر من cout ولكن بالحلقات استطيع كتابة البرنامج في ٥ سطور

ويكون كالاتي

```
#include<iostream>
using namespace std;
int main()
{
int i;
for(i=1;i<=100;i++)
{
cout<<i<<"\t";
}
return 0;
}
```

وبامكاني الحل بشكل اخر

```
#include<iostream>
using namespace std;
int main()
{
for(int i=1;i<=100;i++)
{
cout<<i<<"\t";
}
return 0;
}
```

ملحوظة :-في حالة تنفيذ حلقة for بأكثر من سطر فإننا يجب ان نضع الجمل بين { } لأنها تنفذ السطر التي يليها مباشرة.

حلقة forاللانهائية : for(;;)

تستخدم لتنفيذ الحلقات اللانهائية

مثال:-

#include <iostream.h> #include<conio.h> void main() { for(;;) cout<<"welcome"; getch( ); }	Output:- welcomewelcome welcome

وتستمر الحلقة بالتكرار إلى ما لا نهاية.

عبارتي :-break & continue :-

١. عبارةcontinue: تستخدم مع حلقات forفي حال استبعاد شرط معين من الحلقه ،والمثال التالي سيوضح ذلك:-

#include<iostream.h>
#include<conio.h>
void main()
{
for(int i=0 ;i<=10;i++)
{
If((i==5) ||(i==7))
continue;
cout <<i;
}
getch();
}
	Output:-
	1 2 3 4 6 8 9 10

هنا تم استبعاد العددين5, 7 من الحلقة

٢. عبارةbreak:- تستخدم لأجل توقيف الحلقة عندما يتحقق شرط معين داخل الحلقة for.

مثال:-

#include<iostream.h>
#include<conio.h>
void main()
{
for(int i=0 ;i<=10;i++)
{
if(i==5)
break;
cout <<i;
}
getch();
}
	Output:-
	0 1 2 3 4

هنا توقفت الحلقة عندما وصلت إلى قيمة المتغير i إلي 5.

حلقات for المتداخلة:-Nested for loop:-

في بعض الأحيان، قد تحتوي حلقة for، علي حلقة أخري، وفي هذه الحالة، سيتم تنفيذ الحلقة الداخلية تماماً لكل دورة من دورات الحلقة الخارجية. المثال التالي يوضح ذلك:-

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{   int  i , j;
   for (i=1;i<=3;i++)
{
    cout<<"\n"<< i <<"\t";
    for(j=1;j<=3;j++)
     cout<<j;
 }
     system("PAUSE");
   return EXIT_SUCCESS;
}
```

ملاظة :- عدد مرات تنفيذ الحلقة كاملة = حاصل ضرب عدد مرات تنفيذ الحلقتين

التكرار المتغير:-Variables Loops

حلقة While :-

تستخدم لتكرار جملة أو عدة جمل إلى عدد معين من المرات أو إلى أن يحقق شرط معين. (عدد مرات التكرار غير معروف). فمثلاً إذا كان التكرار يعتمد علي المستخدم، فللمستخدم الحرية في متابعة التكرار أو التوقف، وهذه عملية لا يمكن للمبرمج أن يتنبأ بعدد مرات تكرارها. وأيضاً إذا أردنا البحث عن كلمة في ملف، فنحن لا ندري عدد مرات تكرار الكلمة، بل سنقوم بتكرار قراءة الكلمات من الملف، إلي أن نصل إلي نهاية الملف. والأمثلة كثيرة. لكن هذا لا يمنع من استخدام هذه الحلقات في حالة التكرار المعدود، فطالما تم ربط شرط التكرار بالمتغير العداد للحلقة، يكون التكرار معدود كما في حلقة for.

الصيغة العامة:-

```
While (condition)
{
     statement(s);
}
```

لا تحوي حلقةwhile علي متغير عداد كما في حلقة for لذا عند استخدام حلقة whileيجب مراعاة الاتي:-

1. **بل يجب الإعلان عن المتغير العداد.**
2. **إعطاؤه قيمة ابتدائية قبل البدء في الحلقة.**
3. **زيادته داخل الحلقة.**

مثال: يقوم بطباعة الأعداد من ١ إلى ١٠٠

```
#include<iostream>
using namespace std;
int main()
{
int i=1;
while(i<=100)
{
cout<<i<<"\t";
i++;
}
return 0;
}
```

حلقة do...while:-

وهي أيضا تستخدم لتكرار جملة أو عدة جمل إلى عدد معين من المرات أو إلى حين تحقق شرط معين(عدد مرات التكرار غير معروف).
الصيغة العامة

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

تختلف حلقة do... while، عن حلقة while، في أن شرط التوقف من الحلقة يتم اختباره عند الانتهاء من الحلقة، لذا فإن هذه الحلقة يتم تنفيذها مرة واحدة علي الأقل.

لا تحوي هذه الحلقة متغير عداد، بل يجب الإعلان عن المتغير العداد، وإعطاؤه قيمة ابتدائية قبل البدء في الحلقة، ثم زيادته داخل الحلقة، كما في حلقة while.

مثال: يقوم البرنامج بطباعة الأعداد من ١ إلى ١٠٠

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{ int i=1;
do
{
cout<<i<<"\t";
i++;
} while(i<=100);
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

أمثله في الحلقات

١- اكتب برنامج تدخل عليه درجات ٥ طلاب و والبرنامج يقوم باخراج متوسط درجات الطلاب ؟

والحل في الكود التالي

```
#include<iostream>
using namespace std;
int main()
{
int i=1,coun=0,avg,mark;
for(i=1;i<=5;i++)
{
cout<<"\n enter mark "<<i<<"\t";
cin>>mark;
coun=coun+mark;
}
avg=coun/5;
cout<<"\n avg is :"<<avg<<endl;
return 0;
}
```

٢- اكتب برنامج يقوم بإيجاد حاصل جمع أعداد يتم إدخالها من المستخدم، طالما كان العدد المدخل غير سالب.

```
#include <iostream.h>
#include <conio.h>
void main()
{
   int x,sum=0;
   cout<<"Enter Number: ";
   cin>>x;
   while(x>=0)
   {
      sum+=x;
      cout<<"Enter Number: ";
      cin>>x;
   }
   cout<<"Sum=\t"<<sum;
   getch();
}
```

٣- اكتب برنامج يجمع الأعداد الفردية بين العدد ١ و١٥ ؟

تمارين٣

١ -أكتب برنامج لطباعة مضروب عدد يتم إدخاله من المستخدم.

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{  int  i,n,fact=1;
     cout<<"enter number";
     cin>>n;
     for(i=1;i<=n;i++)
     fact=fact*i;
     cout<<"factorial\t"<<n<<"=";
     cout<<fact<<"\n";
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

٢- اكتب برنامج يجمع الأعداد الفردية بين العدد ١ و١٥ ؟

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
}
    int i=1,sum=0;
for(i=1;i<=15;i+=2)
sum=sum+i;
cout<<"\n sum is :"<<sum<<endl;
    system("PAUSE");
    return EXIT_SUCCESS;
{
```

٣-اكتب برنامج يقوم بإيجاد حاصل جمع أعداد يتم إدخالها من المستخدم، طالما كان العدد المدخل غير سالب.

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
```

```
{   int x,sum=0;
   cout<<"Enter Number: ";
   cin>>x;
   while(x>=0)
   {
      sum+=x;
      cout<<"Enter Number: ";
      cin>>x;
   }
   cout<<"Sum=\t"<<sum<<"\n";
   system("PAUSE");
   return EXIT_SUCCESS;
}
```

٤- أكتب برنامج يطبع الشكل التالي:

```
#include<iostream.h>
 main()
{
int n=10,i,j;
  for(i=0;i<n;i++)
    {
      for(j=0;j<=i;j++)

cout<<"*";
cout<<endl;
}
        }
```

٥- أكتب برنامج يطبع الشكل التالي:

```
#include<iostream.h>
 main()
{
int n=10,i,j;
  for( I = 0; i<n ;i++)
  {

       for( j=i; j < n; j++)


cout<<"*";
cout<<endl;
}
 }
```

المصفوفات Arrays

تعريف:-

هي مجموعة من العناصر المتجانسة (من نفس النوع و نفس الحجم)، و التي تحتل مواقع متجاورة في الذاكرة.

يظهر احتياجنا للمصفوفات عند التعامل مع مجموعة من البيانات تتبع فئة واحده من البيانات .. ومثال على ذلك درجات الطلاب ..

ومن أهم ما يميز المصفوفات هو إمكانية ترتيب المصفوفات حسب ما نريد كما تمكن المصفوفات التعامل مع أكثر من متغير باسم واحد، حيث يتم استخدام اسم المصفوفة + موقع العنصر في المصفوفة للتعامل مع أي عنصر في المصفوفة. وهذا يؤدي الى كفاءة أعلى باستخدام الذاكرة وسرعة أكبر عند تنفيذ البرامج.

أنواع المصفوفات

وهناك نوعين للمصفوفات ..

١- مصفوفة ذات بعد واحد one-dimintional array وهي الشائعة الاستخدام

٢- مصفوفة متعددة الأبعاد multi-dimentional array

الإعلان عن المصفوفات :

الصيغة العامة للإعلان عن المصفوفة:-

```
DATA_TYPE   ARRAY_NAME[SIZE]
```

حيث:

DATA_TYPE: نوع عناصر المصفوفة

ARRAY_NAME: اسم المصفوفة

[SIZE]: عدد عناصر المصفوفة

أمثلة :-

إعلان عن مصفوف من نوع صحيح يحوي ٥٠ عنصر

```
int x [50] ;
```

إعلان عن مصفوفة من نوع حرفي باسم name تحوي ٢٠ عنصر

```
Char name [20];
```

إعطاء عناصر المصفوفة قيم ابتدائية :-

```
int x [3]= {5,9,7} ;
char ch[3]={ 'x','y','z' };
float x[3]={0.5,0.33,0.25}
```

يمكن أن يكون حجم المصفوفة متغير ثابت كالتالي :-

```
int const size=10;
float average[size];
```

ملحوظة :-

- **في c++ يكون الرقم صفر هو موقع أول عنصر في المصفوفة ،فإذا كان لدينا مصفوفة مكونة من Nعنصر يكون عنونتها من**{0-(n-1)}.
- **دائما اسم المصفوفة هو عنون لأول عنصر في المصفوفة،وللتعامل مع أي عنصر من عناصر المصفوفة يتم استخدام اسم المصفوفة +موقع العنصر في المصفوفة والذي يكتب داخل المؤثر الدليلي، مثلا :-**

لوضع القيمة ٥ في العنصر الأول،نكتب:-

```
x[0]=5 ;
x[1]=x[0]+5;
x[2]=x[0]+x[1];
```

وهكذا ..... أي يمكن التعامل مع عناصر المصفوفة كالتعامل مع المتغيرات العادية ،ويمكن استخدام عبارات الإدخال والإخراج مع عناصر المصفوفة ، مثلاً :-

```
cout<<x[0];
```

أيضا:-

```
cin>>x[3]>>x[4];
```

التعامل مع عناصر المصفوفة :-

يتم التعامل مع المصفوفات كالتعامل مع المتغيرات. كما يمكن إجراء جميع العمليات الرياضية وعمليات الإدخال والإخراج علي عناصر المصفوفات.

مثال:

```
#include <iostream.h>
#include <conio.h>
void main()
{
int x[5]={10,20,30,40,50};
x[0]=x[1]+x[0];
x[2]=2*x[1]+x[3];
x[3]=3*x[2]/x[3];
cout<<"x[0]="<<x[0]<<"\n";
cout<<"x[2]="<<x[2]<<"\n";
cout<<"x[3]="<<x[3]<<"\n";
getch();
}
```

```
Output:-
x[0]=30
x[2]=80
x[3]=6
```

استخدام الحلقات مع المصفوفات:-

يمكن التعامل مع المصفوفات باستخدام الحلقات ،وذلك لان عناصر المصفوفة تحتل مواقع متجاورة في الذاكرة ، مما يسهل التعامل وتقليل الأوامر في البرنامج .

مثال: برنامج يقرأ خمسة عناصر من المستخدم يخزنها في مصفوفة ثم يطبعها

```
int x[5];
for (int i =0; i<5 ;i++)
cin>>x[i];
for (int i =0; i<5 ;i++)
 cout <<"x["<<i<<" ]= "<<x[i]<<"\ n" ;
```

مثال :- اكتب برنامج يقرأ ١٠ عناصر في مصفوفة ثم يطبعها معكوسة .

الحل :-

```
#include <iostream.h>
#include <conio.h>
int const size=10;
void main()
{
 int x[size];
  for(int i=0;i<size;i++)
   {
     cout<<i<<"==>";
     cin>>x[i];
   }
   cout<<"the array after reverse :\t";
   for(int i=(size-1);i>=0;i--)
     cout<<"\n"<<"["<<i<<"]\t"<<x[i]<<"\n";
  getch();
```

```
Output :-
0==>10
1==>20
2==>30
3==>40
4==>50
5==>60
6==>70
7==>80
8==>90
9==>100
the array after
reverse :
[9]    100
[8]    90
```

}	[7] 80 [6] 70 [5] 60 [4] 50 [3] 40 [2] 30 [1] 20 [0] 10

مثال:- اكتب برنامج يقرا ١٠ أعداد في مصفوفة ثم يحفظ مربعات الأعداد في مصفوفة أخرى ..
الحل :-

```
#include <iostream.h>
#include <conio.h>
int const size=10;
void main()
{
 int a[size];
 int as[size];
 cout<<"enter 10 number :-\n";
 for(int i=0;i<size;i++)
 {
   cin>>a[i];
 as[i]=a[i]*a[i];
 }
 cout<<"the number with square :-\n";
 for(int i=0;i<size;i++)
 cout<<as[i]<<"\n";
 getch();
 }
```

```
Out put:-
enter 10 number :-
10
20
30
60
40
50
60
50
30
20
the number with square :-
100
400
900
3600
1600
2500
3600
2500
900
400
```

المصفوفات ذات البعدين :-

هي عبارة عن مصفوفة مصفوفات أي انها تحوي عناصر في صورة صفوف وأعمدة وتستخدم للتعامل مع البيانات الجدولية . كما في الشكل التالي :-

- الهدف الأساسي من وجود المصفوفات ذات البعدين هو التعامل مع البيانات الجد وليه .

التعامل مع عناصر المصفوفة ذات البعدين:-

تحتاج إلى دليلين الاول يشير الى رقم الصف والثاني يشير الى موقع العنصر في هذا الصف (العمود) .
الإعلان عن المصفوفة ذات البعدين :-
الصيغة العامة :-

data_type array_name[row_size][column_size];

حيث:
data_type: نوع عناصر المصفوفة
array_name: اسم المصفوفة
[row_size]: عدد الصفوف
[column_size]: عدد الأعمدة

مثلا :-

int x[3][3]; مصفوفة مكونة من ثلاثة صفوف ترقيمه(٠-٢)وثلاثة أعمدة ترقيمه(٠-٢)
int m[4][6];

تحفظ اسماء١٠٠طالب بحيث لا يتجاوز اسم الطالب ٢٠ حرف char name[100][20];

إعطاء عناصر المصفوفة ذات البعدين عناصر ابتدائية :-

امثله :-

```
int x[2][2]={{-10,-20},{50,80}};
int x[2][2]={-10,-20,50,80};
char name[2][3]={'a','b','c','d','e','f'};
```

	0	1	2
0	a	b	c
1	d	e	f

مثال :- اكتب برنامج يقرأ أعداد صحيحة في مصفوفة من ٣ صفوف و ٤ أعمدة ثم يطبع المصفوفة على الشاشة .

الحل:-

```
#include <iostream.h>
#include <conio.h>
void main()
{
 int z[3][4];
 for(int i=0;i<3;i++)
  {
    for(int j=0;j<4;j++)
     {
       cout<<i<<","<<j<<"==>\t";
       cin>> z[i][j];
     }
  }
 cout<<"the array is:\n";
 for (int i=0;i<3;i++)
  {
    for(int j=0;j<4;j++)
    cout<<z[i][j]<<"\t";
    cout<<"\n";
  }
  getch();
 }
```

```
Output:-
0,0==> 1
0,1==> 2
0,2==> 3
0,3==> 4
1,0==> 5
1,1==> 6
1,2==> 7
1,3==> 8
2,0==> 9
2,1==> 10
2,2==> 11
2,3==> 12
the array is:
1     2     3     4
5     6     7     8
9     10    11    12
```

تمرين :-

اكتب برنامج يقرأ مصفوفتينa,b ٣*٣ ثم يوجد حاصل جمع وضرب المصفوفتين .

c=a+b

d=a*b

ضرب المصفوفتين

```
int main()
{
int z[3][3];
int k[3][3];
 for(int i=0;i<3;i++)
  {
    for(int j=0;j<3;j++)
     {
       cout<<i<<","<<j<<"==>\t";
       cin>> z[i][j];
     }}
    cout<<"enter k array\n";
    for(int i=0;i<3;i++)
```

جمع المصفوفتين

```
int main()
{
int z[3][3];
int k[3][3];
 for(int i=0;i<3;i++)
  {
    for(int j=0;j<3;j++)
     {
       cout<<i<<","<<j<<"==>\t";
       cin>> z[i][j];
     }}
    cout<<"enter k array\n";
    for(int i=0;i<3;i++)
```

```
{
  for(int j=0;j<3;j++)
   {
     cout<<i<<","<<j<<"==>\t";
     cin>> k[i][j];
   }}
 cout<<"the sum of tow arrays is:\n";
for (int i=0;i<3;i++)
 {
   for(int j=0;j<3;j++)
   cout<<z[i][j]+k[i][j]<<"\t";
   cout<<"\n";
 }
```

```
{
  for(int j=0;j<3;j++)
   {
     cout<<i<<","<<j<<"==>\t";
     cin>> k[i][j];
   }}
 cout<<"the sum of tow arrays is:\n";
for (int i=0;i<3;i++)
 {
   for(int j=0;j<3;j++)
   cout<<z[i][j]*k[i][j]<<"\t";
   cout<<"\n";
 }
```

المؤشرات:-(pointers)

المؤشر هو عبارة عن متغير يحمل عنوان متغير آخر موجود في الذاكرة.
وتستخدم المؤشرات في التطبيقات التي نحتاج فيها إلى حجز وإلغاء حجز الذاكرة ديناميكيا (أثناء تنفيذ البرنامج).

الإعلان عن المؤشرات في c++:-

الصيغة العامة :-

```
data_type *  pointer_name;
data_type *pointer_name;
```

أمثلة :-

```
int *p ;
```

P عبارة عن مؤشر من نوع صحيح

```
char *  c;
```

c عبارة عن مؤشر يشير الى متغير من نوع حرفي .

استخدام المؤشرات :-

قبل استخدام أي مؤشر يجب أن يعطى قيمة ابتدائية ،وذلك باستخدام مؤثر المرجع (&) كالآتي :-

```
int x=10;
int *p=&x;
```

او

```
int *p;
p=&x;
```

العمليات المعرفة على المؤشرات :-

هنالك عمليتان معرفتان على المؤشرات :-

١. مرجع(referencing):-

جعل المؤشر يشير إلى عنوان متغير موجود.

مثلا :-

```
int x=10;
int *p=&x;
```

٢. استرجاع(dereferencing):-

استرجاع محتوى العنوان الذي يشير إليه المؤشر.

مثلا :-

```
int x=10;
int *p=&x;
cout<<*p;➔10
```

الدوال:

الدالة :-

هي عبارة عن برنامج فرعي يتم تعريفه قبل البرنامج الرئيسي ، ثم يتم استدعائه داخل البرنـامج الرئيسي ليقوم بوظيفة معينة .

مزايا الدوال :

١. **تساعد على تقسيم المشكلة إلى مجموعة من المشاكل الصغيرة والتي يسهل برمجتها.**
٢. **تساعد على سهولة الفهم.**
٣. **تساعد علي إعادة صياغة البرنامج.**
٤. **تساعد على اكتشاف الأخطاء.**
٥. **عدم التكرار.**

الصيغة العامة لكتابة الدالة :-

```
(1)Data type  (2) Functions name (3)(parameterlist)
{
  // Functions body (4)
}
```

(١) نوع رجوع بيانات الدالة.
(٢) اسم الدالة.
(٣) قائمة الوسائط التي تستقبلها الدالة.
(٤) جسم الدالة (التعليمات).

الأمثلة التالية توضح إعلانات صحيحة لدوال:-

```
1.void display ( );
2.void add (int , int );
3.void print (char,int,float);
4.int add (int ,int) ;
5.float average (int[ ],int);
6.void set_name (char*);
7.char* get_name( );
8.void swap (int&,int&);
```

استدعاء الدالة (Functions call) :

ويتم استدعاء الدالة باسمها مع مراعاة نوع وعدد الوسائط التي تستقبلها.

مثال:-

اكتب دالة تقوم باستقبال قيمة صحيحة ، ثم تزيد عليها 5 وتطبع الناتج علـــى الشاشة ؟

الحل:-

```
#include<iostream.h>
#include<conio.h>
 void plus5 (int);
 void main ()
{
 int x = 20;
 plus5(70);
 plus5(x);
 plus5(x+40);
 getch();
}
void plus5(int x)
```

```
Outp

75
25
65
```

```
{
 cout <<(x + 5)<<'\n';
}
```

إرجاع قيمة من الدالة (Returning value) :-

إرجاع قيمة من الدالة إلي البرنامج المنادي يتم باستخدام جملة return داخل الدالة. وجملة return تفيد الأتي:

١. يتم إرجاع قيمة المتغير أو التعبير الموجود في return .
٢. يتم إيقاف تنفيذ الدالة، والرجوع مباشرة إلى البرنامج المنادي.

مثال:-

عدل الدالة في المثال (٢) بحيث ترجع قيمة صحيحة ؟

الحل:-

```
#include<iostream.h>
#include<conio.h>
int plus5(int);
void main ( )
{
 int x = 20, z;
 z =  plus5 (x);
 cout <<z<<endl ;
//another valid call
cout<<plus5(x)<<endl;
//also this is a valid call
cout<<plus5(plus5(x))<<endl;
getch();
}
int plus5 (int x)
{
 return (x + 5) ;
}
```

Output:-
25
25
30

مثال :- اكتب دالة تقوم بإرجاع مكعب عدد صحيح يتم إدخاله من قبل المستخدم ..؟

الحل :-

```
#include<iostream.h>
#include<conio.h>
int cube(int);
int main()
{
int x;
cout<<"enter x:\t";
cin>>x;
int result=cube(x);
cout<<x<<"^3="<<result<<endl;
getch();
}
int cube(int n)
{
  return (n*n*n);
}
```

المتغيرات المحلية والمتغيرات العامة (local variables && global variables):-

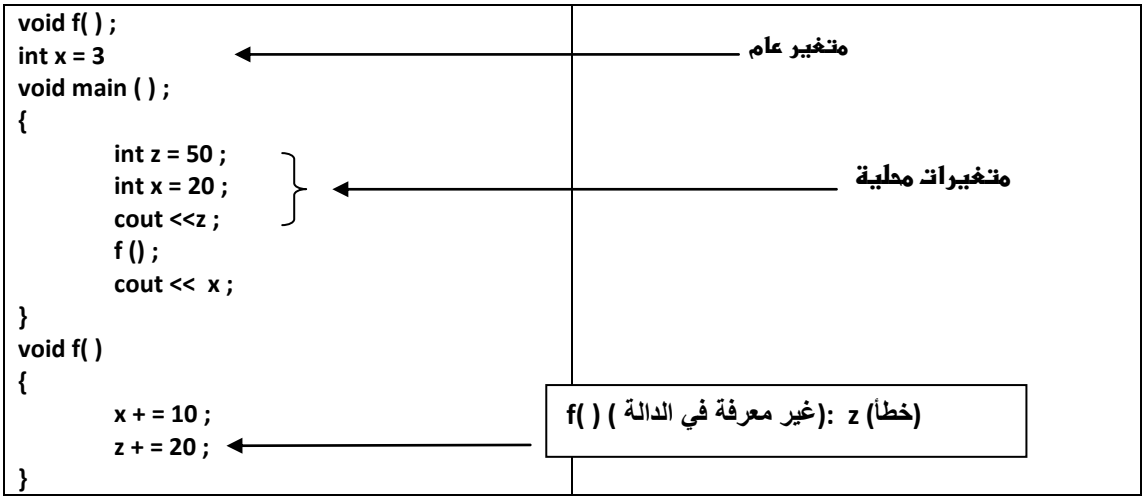
المتغير العام:- Global variables

هو عبارة عن متغير يتم الإعلان عنه على مستوى البرنامج، ويمكن الوصول إليه من أي دالة أخرى معرفة داخل البرنامج، وينتهي مداه بانتهاء البرنامج .

المتغير المحلي :Local variables

هو عبارة عن متغير يتم الإعلان عنه داخل نطاق دالة معينة، او على مستوى block { } معين ، ولا يمكن الوصول إليه من أي دالة أخرى في البرنامج ، وينتهي تعريفه بانتهاء المجال المعرف فيه.

مثال :-



مثال:-

#include<iostream.h>
#include<conio.h>
int y=30;
void main()
{
int y=10;
cout<<"y is local:"<<y<<endl;
cout<<"y is global:"<<::y<<endl;
getch();
}
	Output:-
	y is local:10
	y is global:30

ملحوظة: في المثال السابق

- نلاحظ ان ناتج ال y في الاولى =١٠ وذلك لان y الخاصة قد حجبت yالعامة على مستوى الدالة التي تقع فيها y الخاصة ..
- نلاحظ ان ناتج ال y في العبارة الثانية =٣٠ وذلك لاننا استخدمنا المؤثر ::(scope resolution operator) قبل yوهذا المؤثر وظيفته هو يعيدنا الى قيمة yالعامة قبل ال الدالة الرئيسية. void .