كلية العلوم

القسم : الفيزياء

السنة : الثانية

المادة : لغات البرمجة١

المحاضرة : الاولى والثانية/نظري/

{{ مكتبة A to Z }}

Facebook Group : مكتبة A to Z

الفصل الأول

الخوارزمية وأنواعها

Algorithm and its types

1-1 المشكلة

بزيادة فهم المشكلة يزداد تبعاً له وضوح تفصيلات وأبعاد المشكلة، وبالتالي تصبح المشكلة أكثر تفصيلاً وثباتاً ووضوحاً، هذا يوضح القاعدة الثانية لديكارت والتي تنص على:

- **قاعدة 2**

" يجب أن تتم محاولة تقسيم المشكلة إلى أجزاء بسيطة وغير معتمدة على بعضها البعض ثمّ يجب التركيز على كلّ جزء على حدى".

قاعدة 2أ

يجب أن تتم محاولة تقسيم المشكلة إلى مجموعة مشاكل فرعية بسيطة ومتتابعة، بحيث نحصل على الحل الكامل للمشكلة الأصلية بحل المشاكل الفرعية البسيطة الواحدة تلو الأخرى.

قاعدة 2 ب

إذا كانت المشكلة تتضمن بعض العمليات التي يعاد تكرارها، حاول عزل العمليات التي لا تتطلب الإعادة من تلك التي تتطلب الإعادة.

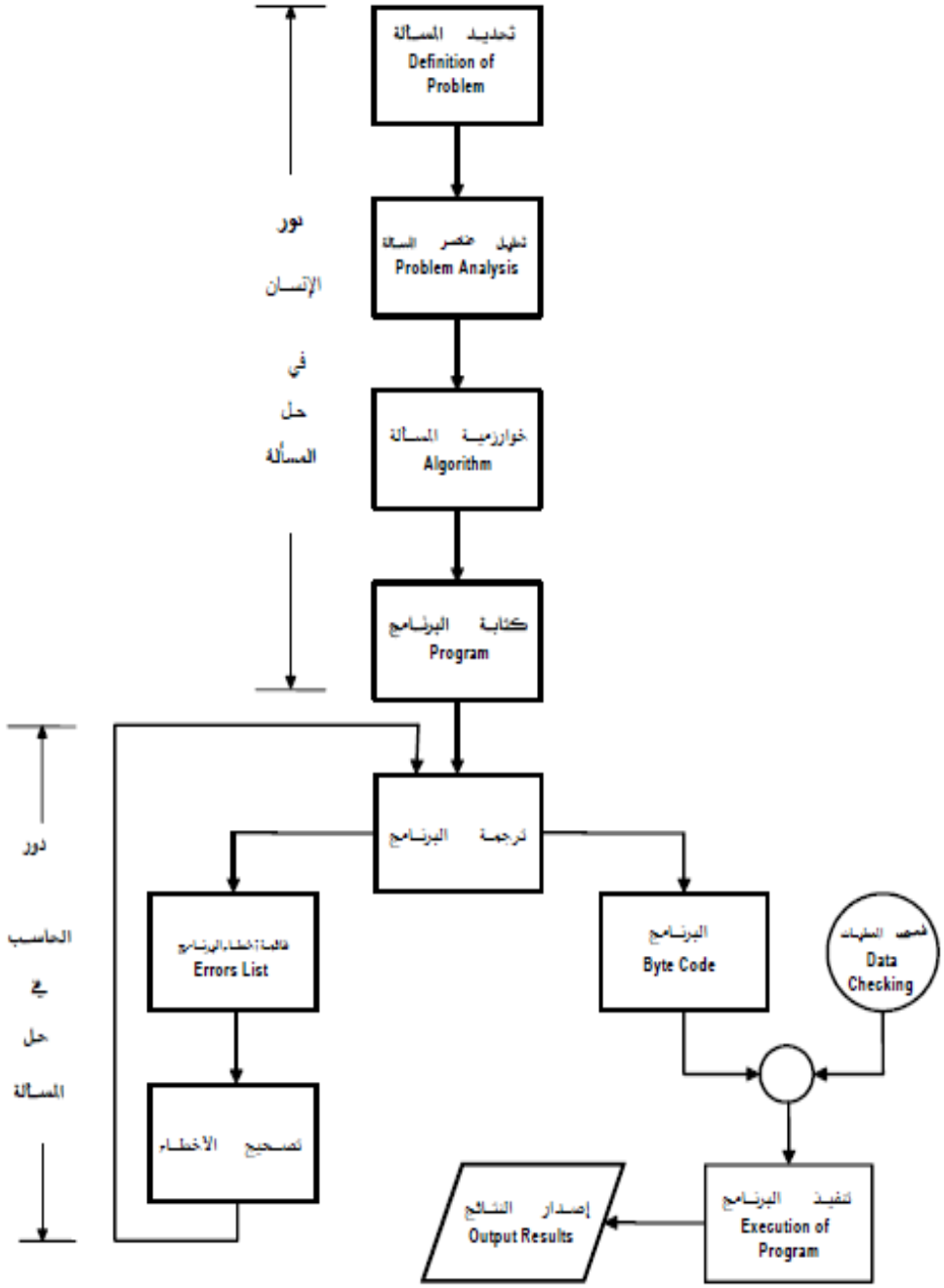
1-2- الخوارزمية والكود الزائف Algorithm and Pseudo Code

بعد أن استعرضنا خطوات التفكير لحلّ أية مسألة برمجية وقبل أن ندخل في تفاصيل كتابة الخوارزمية لحل المسألة، نقول إن الحل يمر بمرحلتين:

المرحلة الأولى

هذه المرحلة تمثل دور الإنسان في حل المسألة وتتكّون من عدة خطوات تعرضنا لها فيما سبق ونجملها فيما يأتي:

- تحديد معالم المسألة
- تحليل عناصرها، وذلك بمعرفة معطياتها، والهدف الأساسي لها، وأهم النتائج المطلوبة منها، وما هي الصورة المراد عرض النتائج فيها، وكذلك صورة تقديم المعطيات.
- البحث والتفكير في طريقة حلّ المسألة
- تدوين الحل في خطوات متسلسلة متعاقبة، يعبر عنها باللغة العادية محكومة بالمنطق الرياضي. هذه الخطوات في مجموعها تسمى بالخوارزم Algorithm، كما يمكن تمثيل هذه الخطوات والارتباط فيما بينها بما يعرف بمخطط التدفق Flowchart، وذلك لكي تساعد في تسلسل المنطق العام لحل المسألة– وسوف نتعرض بالتفصيل لشرح كلّ من الخوارزمية ومخططات التدفق لاحقاً.


تحديد المسألة
Definition of Problem
تحليل عناصر المسألة
Problem Analysis
خوارزمية المسألة
Algorithm
كتابة البرنامج
Program
دور الإنسان في حل المسألة
ترجمة البرنامج
قائمة أخطاء البرنامج
Errors List
تصحيح الأخطاء
البرنامج
Byte Code
فحص المعطيات
Data Checking
تنفيذ البرنامج
Execution of Program
إصدار النتائج
Output Results
دور الحاسب في حل المسألة

- **المرحلة الثانية**

هذه المرحلة تمثل دور الحاسب نفسه في حلّ المسألة، والتي تبدأ بترجمة البرنامج المكتوب بلغة المستوى العالي الى لغة الآلة بواسطة المترجم Compiler، ومن ثم يقوم بحفظ البرنامج في الصورة الجديدة حتى يتم تنفيذه بعد ذلك لإخراج النتائج إلى الوسط الخارجي، ليقوم المستخدم بالاستفادة منها بالشكل الذي يريده وذلك عند عدم وجود أخطاء في البرنامج. أما في حالة وجود أخطاء في البرنامج فإنه يجب تصحيح هذه الأخطاء أولاً ثم تعاد الترجمة مرة ثانية وهكذا حتى نحصل على برنامج بدون أخطاء ثم بعد ذلك يتم تنفيذ البرنامج.

1-3- الخوارزميات Algorithms

تستخدم كلمة الخوارزمية، على نطاق واسع، في علوم الرياضيات والحاسب، الآن حيث تعرف بأنها:

مجموعة الخطوات (التعليمات) المرتبة، لتنفيذ عملية حسابية، أو منطقية، أو غيرها بشكل تتابعي متسلسل ومنظم.

1-4- مخططات التدفق Flow Charts

تستخدم مخططات التدفق في بيان خطوات حلّ المسألة وكيفية ارتباطها ببعض، باستخدام رموز اصطلاحية لتوضيح خطوات الحلّ.

تعتبر رسوم مخططات التدفق المستعملة في تصميم حلول بعض المسائل، مرجعاً، في حلّ مسائل أخرى مشابهة، ومفتاحاً لحلّ مسائل جديدة لها علاقة مع المسائل القديمة المحلولة، فتُشبّه رسوم مخططات التدفق، والحالة هذه، بالرسوم التي يضعها المهندس المعماري عند تصميمه بيتاً أو عمارة، أو مسجداً...الخ.

1- للبداية والنهاية start / stop	
2- للإدخال والإخراج input / output	
3- للعمليات الحاسوبية computer process	
4- الشرط والتقرير (القرار) decision	
5- لاستدعاء البرنامج الفرعي call subroutine	
6- لاتجاه سير البرنامج flow line	
7- لنقاط التوصيل والربط connector	
8- مستند document	

مثال:

اكتب باستخدام المخططات التدفقية خوارزمية حساب مساحة ومحيط مستطيل أطوال أضلاعه A و B.

الحل: يوضح الشكل 1-1 خطوات الخوارزمية التي يمكن استخدامها للحل ممثلةً بالمخطط التدفقي التتابعي والتي يمكن كتابتها باستخدام اللغة الطبيعية كما يلي:

1-البداية.

2-أدخل طول وعرض المستطيل.

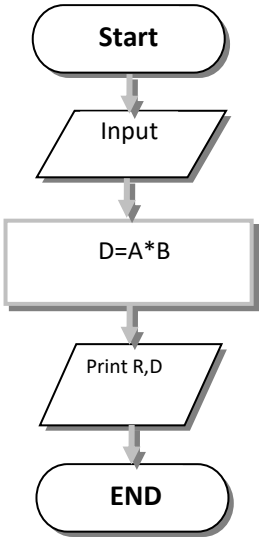
3-احسب:

المساحة = الطول × العرض

المحيط = (الطول + العرض) × 2

4-اطبع (أخرج) قيمة المساحة والمحيط.

5-توقف (النهاية).



الشكل 1-1 خوارزمية حساب مساحة ومحيط مستطيل ممثلةً بالمخطط التدفقي التتابعي

الفصل الثاني

مدخل إلى البرمجة بلغة C++

Introduction to the programing using C++

2-1- مقدمة

يتضمن هذا الفصل بعض تعليمات وبنى معطيات لغة C++ المصنفة تحت اللغات البرمجية عالية المستوى خلاف للغات **منخفضة المستوى**: لغة **الآلة** ولغة **التجميع** والتي سيتم الاستفادة منها في تنفيذ الخوارزميات المشروحة.

- لغة الآلة Machine Language: وهي تتألف من مجموعة من الأصفار والواحدات التي تعطي الأوامر إلى الحاسب من أجل تنفيذ تعليماته كل تعليمة على حده، وبالتالي تصبح صعبة على المبرمجين مما جعلهم يفكرون بتجميع عدة تعليمات بتعليمه واحدة، مما أنتج لغة التجميع.

لغة التجميع :**Assembly Language** حيث تتضمن التعليمة الواحدة أكثر من تعليمة بلغة الآلة ولكي يفهما الحاسب كان لابد من المترجم أو المفسر الذي يقوم بتبديلها إلى لغة الآلة.

- **اللغات عالية المستوى** **High-Level Language**: تتضمن التعليمة الواحدة أكثر من تعليمة بلغة الآلة وكذلك تكون اللغات عالية المستوى قابلة للتطبيق على معظم أجهزة الكومبيوتر كما تمتاز هذه اللغات بالبساطة من حيث الفهم من قبل المبرمج وهي تحتاج بالطبع إلى مترجمات أو مفسرات.

يوجد طريقتان للترجمة من لغة التجميع واللغات عالية المستوى إلى لغة الآلة.

1- **المترجم: Compiler** يقوم بترجمة البرامج الموجودة بلغة عالية المستوى أو لغة التجميع إلى لغة الآلة وبعدها يمكن تنفيذها، حيث يدعى البرنامج بلغة عالية المستوى ببرنامج المصدر source program والبرنامج بلغة الآلة ببرنامج الهدف object program وتحتاج المترجمات إلى زمن طويل في الترجمة وزمن قصير في التنفيذ.

2- **المفسر Interpreter:** يفسر كل سطر بعد الانتقال إلى السطر الذي يليه ويحتاج إلى زمن قصير في التفسير وطويل في التنفيذ لذا تستخدم المفسرات عند تطوير البرامج وبعدها يتم ترجمتها.

تعتبر لغة C++ لغة برمجية للأغراض العامة حيث تتألف من كلمات رئيسية key words مثل while، do، for، else، if وهي تشبه لغة Pascal أو Fortran.....الخ، كما ويمكن أن تستخدم كلغة منخفضة المستوى نظراً لاحتوائها على سمات إضافية تمكن ذلك حيث يمكن أن تستخدم في برامج نظم التشغيل.

2- 2 - أساسيات لغة C++ (Essences of C++)

يهتم الفصل بالعناصر الأساسية المستخدمة في إعداد عبارات بسيطة، التي تشمل فئة رموز لغة C++ والمعرفات، والكلمات الرئيسية "المحجوزة"، وأنواع البيانات، الثوابت، المتغيرات، المنظومات، والتوضيحات، والتعبيرات، والعبارات.

2-2-1 - فئة رموز لغة C++

تتألف أبجدية لغة C++ من:

- المحارف الكبيرة والصغيرة A — Z (تميز لغة C و C++ بين المحارف الصغيرة والكبيرة).

– الأرقام من 0 إلى 9.

– بعض الرموز الخاصة كعناصر لبناء هيكل البرنامج، ونسرد بعض الرموز الخاصة:

! * + \ " < # (= | {) % ≈ ; } . ^ [! ، ? & -} '. (blank)

كما وتوجد رموز أخرى منها @، $، وتستخدم لغة c++ العديد من بدائل الخلط بين هذه الرموز مثل(\b،\n،\L) الحركة الخلفية لمسافة واحدة back space، السطر الجديد، والجدول الرئيسي على التوالي وهذا ما يعرف بتتابعات أو تسلسلات الهروب، وتسلل الهروب يمثل رمزاً فردياً واحداً.

2– 2 –2– المعرفات والكلمات المحجوزة Identifiers and Key words

المعرفات ما هي إلا أسماء عناصر البرنامج المختلفة، مثل المتغيرات والتوابع والمصفوفات، وتشكل المعرفات بواسطة أبجدية اللغة من محارف وأرقام إلا أن ذلك يشترط أن يكون الرمز الأول في المعرف محرف ويجوز أن يكون تسطيره سفلية، وبالتالي من الخطأ البدء برقم:

- ألا يحوي على الرموز الخاصة.
- كذلك يجب ألا يحتوي المعرف على فراغ
- ألا يكون كلمة محجوزة.

تسمح لغة c++ بطول معرفات حسب الرغبة ويصل في بعض أجياله حتى 31 محرف.

مثال عن المعرفات الصحيحة:

table_ name _temperature sum_1 y12 X

مثال عن المعرفات الخاطئة:

4th معرف خاطئ لأنه يبدأ بعدد وليس بحرف، "x" معرف خاطئ لأنه يبدأ برمز خاطئ، order –on معرف خاطئ لأن الإشارة – رمز خاص غير مسموحة .

وسنورد في ما يلي بعض الكلمات المحجوزة بلغة C وطبعاً هي الكلمات التي لايسمح باستخدامها كمعرفات:

```
auto          break       case          char          const
continue default        do       doubl        else       enum
long  extern  float  for goto  if  int register      return
short         signed          sizeof            static
struct switch       typedef      union     while volatile
unsigned    voil
```

وفي لغة C++ نجد أيضاً:

```
asm           catch       class     delete          friend
inline new  operator    private      public         template
this protected    throw    try virtual
```

2-2-3- أنواع البيانات Data types

يوجد في لغة C++ أنواع عدة من البيانات وتختلف عن بعضها البعض بعدد البايتات التي تخصص لها في ذاكرة الحاسب وفي النوع الذي تمثله، ويختلف هذا العدد من مترجم لأخر.

إن نوع المعطيات int ومشتقاته long int و short int يسمح بتخزين معطيات من النوع الصحيح، أما float وdouble فهي تسمح بتخزين معطيات من النوع الحقيقي، حيث يسمى المتغير من النوع double بالمتغير الحقيقي ذي الدقة المضاعفة أما char فيسمح بتخزين معطيات محرفيه.

2-2-4- الثوابت constants

توجد أربعة أنواع من الثوابت: صحيحة، حقيقية، محرفيه، string "متعددة المحارف".

الثوابت الصحيحة والحقيقية تتميز بما يلي:

1-عدم وجود فراغات ضمنها.

2-يمكن أن تُسبق بإشارة.

3-لا يمكن أن تتعدى قيمة ثابتة أو أن تقل عن حد معين، أي لها حد علوي وسفلي.

أ-الثوابت الصحيحة: Integer constants

هي أعداد صحيحة يمكن أن تمثل بالنظام العشري والثماني والسداسي عشر.

عند تمثيلها بالنظام العشري نجد أن الثابت يتكون من الأرقام 9...0 ، المرتبة الأخيرة من اليسار غير صفرية إذا كان العدد اكبر من رقم واحد.

أما عندما يمثل الثابت بالنظام الثماني نجد أنه يتكون من الأرقام من 0 إلى 7 ويسبق بـ 0 (zero character) مثلاً:

```
00      01    0710     077
```

أما عندما يمثل الثابت بالنظام السداسي عشر نجد أنه يتكون من الأرقام من 0 إلى 9والمحارف A،B،C،D،E،F ويسبق بـ 0x مثلاً:

```
0x0         0x1          0x7FF
```

ب–الثوابت الصحيحة الطويلة "بدون إشارة"

يمكن أن يعرف الثابت الصحيح بدون إشارة unsigned بإضافة الحرف U صغير أو كبير إلى الطرف اليساري للعدد ويمكن تعريف ثابت صحيح طويل long بإضافة الحرف L إلى ذلك الطرف.

ويمكن تعريف ثابت صحيح طويل بدون unsigned long إشارة بإضافة UL ويطبق ذلك على الثماني والسداسي عشر أيضاً، أي كافة أنواع التمثيل العددي.

ج– الثوابت الحقيقية **Real constants:**

يحوي العدد أس عشري أو نقطة عشرية أو كليهما ونذكر بعض الأمثلة.

```
0.      1.      0.7      320.1      1.66E+8      166E-2
3E-8
```

يجب أن يكون الأس عدد صحيح لا يملك فراغات ولا فاصلة عشرية أو عادية.

مثال: يمكن تمثيل القيمة 3.10^5 بعدة طرق.

```
3e5     3E5     3e+5     300000      .3e6     300e3     30.E4
```

د– الثوابت المحرفية **Character constants:**

هي رمز واحد بين فاصلتين علويتين ´A´ ،´a´ ،´´ ´$´ ويمكن لكل حرف أن يأخذ قيمته من الجدول المعياري.

American Standard code for Information Interchange (ASCII)

يعتبر جدول ASCII كمقياس لشفرات الرموز الفردية حيث يتم حجز 1byte لكل رمز 2^7=128 ويمكن لبعض الأجهزة أن تستخدم فئة رموز الشفرة الثنائية الموسعة للمعلومات العشرية والتي تحدد فيها شفرة عددية لكل رمز فردي له خليط من 8bit خاصة به وتدعى:

Extended binary coded decimal information code (Ebcdic)

ونذكر في الجدول (2-1) تسلسل الهرب لبعض الرموز التحكمية والتي تبدأ بالفاصلة المائلة \ التي تدعى بمحرف الهروب escape character والتي تدل على وجود عمل خاص يجب فعله ثم يليها حرف أو رمز.

الجدول(2-1) سلاسل الهروب الشائعة

الفعل	تسلسل الهرب	ASCII
لسماع الجرص Alert(beep)	\a	007
للانتقال إلى الخلف	\b	008
للانتقال جدول أفقي	\t	009
للانتقال جدول رأسي	\v	011
للانتقال إلى سطر جديد	\n	010
تغذية صيغة الورقة	\f	012
عودة العربة إلى بداية السطر الحالي carriage return	\r	013
لطباعة علامة تنصيص مزدوجة	\"	034
لطباعة علامة تنصيص مفردة	\'	039
لطباعة علامة استفهام	\?	063
لطباعة شرطة مائلة للخلف	\ /	092
لطباعة صفر	\0	000

والرمز \0 له تمثيل خاص ويمثل نهاية السلسلة وإن '\0 ' لا يكافئ .'0'

2-2-5- المتغيرات variables

المتغير يمثل قيمة معطيات بسيطة ضمن البرنامج، عددية أو حرفية وتملك هذه المتغيرات قيماً وأنواع معطيات مختلفة، ولكن المتغير الواحد يملك نوع واحد من المعطيات وقيم مختلفة (ولكن قيمة واحدة في اللحظة الواحدة) منه وتعرف وفق البرنامج كما يلي:

```
int    a ،b ،e ;          char    d;
```

وهنا يجب نسب القيم الصحيحة العددية للمتحولات الثلاث الأولى ونسب قيمة محرفية إلى المتغير d ضمن البرنامج ويمكن تغيير القيم العددية والمحرفية ضمن البرنامج مع المحافظة على النوع.

2-2-6- المصفوفات arrays

عبارة عن تجمع عدد من القيم المتساوية النوع ضمن نفس الاسم، والعنصر الواحد هو عنصر مصفوفة وتميز العناصر عن بعضها بعضاً بتمايز الدليل للمصفوفة، والذي يبدأ من 0 وينتهي مع n-1 إذا احتوت المنظومة n عنصر ويمكن أن تتضمن أعداد صحيحة أو حقيقية أو محرفية وأن تكون ببعد واحد أو أكثر.

والمصفوفة المحرفية ببعد واحد هي سلسلة من المحارف خلف بعضها البعض وتنتهي مع \0 وبالتالي لتخزين n حرف يلزمنا n+1 مكان لتخزين الرمز الأخير \ 0 ويوضع الدليل بين قوسين مربعين:

```
A[n-1].....A[2] A[1]
```

0	1	2	3	4	5	6	7
T	A	R	T	O	U	S	\ 0

وبفرض تم تخزين TARTOUS فإن ذلك يتطلب ثمان أماكن تخزين.

ويجب التصريح عن المتغيرات والمصفوفات قبل استخدامها ضمن البرنامج وذلك لتحديد حجمها لكي يقوم المطابق بالحجز اللازم والكافي مثال ذلك:

```
int  a ،b ،e ;      float a1،b1،e1;        char ax ،bx[10];
```

حيث عرف في الجملة الأولى a، b ،e ثلاث أعداد صحيحة والجملة الثانية a1،b1،e1 ثلاث متغيرات حقيقة والجملة الأخيرة ax،bx[10] متغير محرفي وسلسلة محرفية 10 حروف.

2-2-7- التعابير expressions

تملك قيمة فردية عددية، محرفية أو عنصر مصفوفة أو إشارة تابع ويمكن أن تكون خليط من ذلك تتصل مع بعضها بمعاملات ويمكن أن تكون نتيجة التعبير قيمة عددية أو منطقية تملك القيمة false= 0، true = 1 .

وهذا ما يدعى بالمؤثر الأحادي نظراً لتأثيره على عنصر وحيد وسوف نجد المزيد من أنواع المؤثرات والعبارات في فقرات لاحقة.

2-2-8- العبارات statements

يوجد نوعين للعبارات:

أ-عبارة بسيطة: simple statement

وهي عبارة تنتهي بـ ; حيث أن:

```
a=3;
c= a+ b ;
```

العبارة الأولى لنسب القيمة 3 إلى المكان المخصص للمتغير a والثانية لنسب مجموع قيمتي المتغيرين a، b للمكان المخصص للمتغير c.

ملاحظة: عند وجود ; مفردة على السطر دون أي شئ هذا يعني أنها عبارة فارغة.

ب-عبارة مركبة: complex statement

تتألف من عدة عبارات بسيطة يوضع في بدايتها } وفي نهايتها { ولا تملك ; بعد قوس الإغلاق، وتستخدم عند وجود عدة عبارات تتبع لنفس الحالة وتصبح عندها العبارة المركبة وكأنها عبارة بسيطة.

هنا عبارة مركبة من أربع عبارات بسيطة ويستمر تنفيذ العبارة المركبة طالما أن count لم تصبح أكبر من n ويجب أن تزاد قيمة count ضمن العبارة لكي تصبح أكبر.

2-2-9- الثوابت الرمزية symbolic constants

الثابت الرمزي: هو اسم يعوض به عن تسلسل من الرموز التي يمكن أن تمثل ثابتاً عددياً أو ثابتاً حرفياً أو ثابت سلسلة، وعليه يسمح الثابت الرمزي بظهور اسم بدل من ثابت عددي أو ثابت محرفي أو ثابت سلسلة ويستبدل كل وجود للثابت ضمن البرنامج بتسلسل الرموز الخاص به.

تعرف الثوابت الرمزية في بداية البرنامج وبعدها تظهر في أي مكان نحتاج إليها وتعرف كما يلي: #define identifier value

2-3- معاملات لغة C++ Operators of

تستخدم المعاملات الأحادية، والحسابية، والعلائقية، والمنطقية، ومعاملات التخصيص أو النسب، والشرطية في تكوين التعابير أو العبارات البرمجية وتؤثر المعاملات على العوامل أو العناصر وقد تتطلب عامل أو أكثر.

2-3-1- العمليات الحسابية Arithmetic Operators: العمليات الحسابية المعروفة كالجمع Addition (+) والطرح Subtraction(-) والضرب Multiplication(*) والقسمة Division (/) وباقي القسمة Module (%).

تستخدم المعاملات الحسابية عاملين وفي حالتي القسمة وباقي القسمة يجب أن يكون العامل الثاني مخالفاً للصفر، كما أن عاملي باقي القسمة يجب أن يكونا صحيحين.

6- أولويات التنفيذ هي:

1- الأقواس Parentheses

2- (% / *) Modulus، Division،Multiplication

3- (- +) Subtraction،Addition

وإذا تساوت الأولويات يتم التنفيذ من اليسار باتجاه اليمين.

2-3-2- المعاملات الأحادية unary operators

تتعامل هذه المعاملات مع عامل واحد وقد تسبق العامل عادةً أو ربما تكون خلفه في بعض الحالات، أكثر هذه المعاملات شيوعاً هو المؤثر السالب، ويسبق الأعداد والثوابت العددية والتعبيرات الحسابية.

كما ويوجد مؤثر زيادة العداد ++ ومؤثر النقصان-- وهما يؤثران بالزيادة أو النقصان بالقيمة 1 وقد تسبق العامل وقد تليه وسيتم شرحها لاحقاً.

```
m → 5 ;     m ++ → 6 ;
n → 3 ;     ++ n → 4 ;
q → 4 ;     -- q → 3 ;
p → 5 ;      p -- → 4 ;
```

والمعاملات الأحادية تملك أولوية على المؤثرات الحسابية.

2-3-3- المعاملات العلائقية Relational operators

هذه المعاملات هي التي يتم من خلالها المقارنات :

الرمز الرياضي	الرمز في C++	التسمية
>	>	أكبر Greater than
<	<	أصغر Less than
>=	≥	أكبر يساوي Greater or equal than
<=	≤	أصغر يساوي Less or equal than
==	=	يساوي Equal
!=	≠	لا يساوي not equal،Different

والمعاملات الستة لها عاملين وتكون نتيجة تطبيقها على عواملها إما true (1) أو false (0).

وقد تكون العوامل جمل أو تعابير تعطي في النهاية إلى قيم منطقية مثلاً:

```
( i+f ) <= 10 ;     c >= 10*( i+f ) ;
```

2-3-4-المعاملات المنطقية Logical operators

كما وتملك لغة C++ التابعين المنطقيين or، (||) و and (&&).

وبفرض أن هذين التابعين يطبقان على معاملين a،b عندئذ يكون جدول الحقيقة لهما وفق ما هو مبين في الجداول (2-2 و 2-3):

الجدول (2-2) يوضح عمل OR

a	b	a||b
0	0	0
0	1	1
1	0	1
1	1	1

الجدول (2-3) يوضح عمل and

a	b	a&&b
0	0	0
0	1	0
1	0	0
1	1	1

وهنا && لها أولية أكبر من || وتقع بعد المساواة وعدم المساواة.

بالإضافة للمعاملين ||، && هناك معامل النفي الأحادي not (!) وهو يقوم بنفي التعبير المنطقي وله أولية على المعاملات الأحادية.

إن لتنفيذ العبارات أولوية في التنفيذ ما بين العمليات والجدول (2-4) يوضح أولوية المعاملات:

2-3-5- عوامل التخصيص أو النسب Assignment operators

إن أكثر عوامل التخصيص شيوعاً هو = وله الشكل:

```
identifier  =expression;
```

إن identifier هو متغير بصورة عامة أما expression فهو متغير أو تعبير أو خليط منهما، مثلاً:

```
a=3;          delta =0.001 ;          area =length *width;
```

إن = كما أسلفنا يدعى بعامل النسب أو التخصيص، أما = = فيدعى بالتساوي ويختبر قيماً إذا كانت القيمة في اليمين تساوي القيمة في اليسار ونتيجته منطق .

وعندما يكون طرفي التخصيص من أنواع معطيات مختلفة، يتحول ما هو في اليمين إلى ما يماثل ما هو في اليسار وبالتالي يمكن أن يحذف الجزء الكسري من القيمة الحقيقة، أو يتم تقريب القيمة المزدوجة الدقة، وكذلك القيمة الصحيحة إلىshortint.....

وهذا يكافىء تماماً الصيغة:

```
expression1 = expression1 + expression2 ;
```

2-3-6- المعامل الشرطي

```
Conditional operator
```

ويرمز له

```
expression1 ? expression2 : expression3
```

هنا يتم التساؤل هل قيمة`true expression1` أم `false` فإن كانت `true` سيتم أخذ القيمة `expression2` وإذا كانت تساوي `false` سيتم أخذ القيمة الأخيرة `expression3` ومثال ذلك.

```
i=( i < 0 )  ? 0 : 100
```

إذا كانت `i<0` أي التعبير صح (1) سوف يتم إسناد القيمة `0` للمتغير `i` وغير ذلك سيتم إسناد القيمة `100` له.

```
min = ( f < g ) ? f : g
```

إذا كانت `f` أصغر من `g` سيكون `min = f` وإلا `min = g`.

والمعامل الشرطي يقع قبل عامل التخصيص وبعد المعاملات المنطقية.

2-4-التوابع function

هي مجموعة من العبارات البرمجية المجمعة ضمن مقطع برمجي واحد يملك اسماً وموجود مع التوابع الأخرى ضمن مكتبة خاصة بلغة البرمجة.

ينادى التابع من برنامج آخر أو تابع آخر ويعيد هذا التابع قيمة واحدة قد تكون عددية، محرفية، منطقية.....وقد لا يعيد شيء.

وتوجد توابع كثيرة منها للتعامل مع إدخال وإخراج البيانات من وحدة الدخل القياسية أو من الملفات.... وتوابع للتعامل مع الشرائط المحرفية وأخرى للحسابات أو المثلثات..

ينادى التابع من خلال اسم التابع يليه مجموعة من المتغيرات تتوضع بين قوسين وتفصل عن بعضها بعضاً بفواصل ويمكن أن تكون ثوابت، أسماء متغيرات، تعبيرات أكثر تعقيداً ويمكن أن توضع الأقواس حتى بدون أية عناصر ضمنها.

وتوجد العديد من التوابع القياسية المتوفرة لدى المترجم كما ويمكن كتابة توابع من قبل المستخدم وضمها إلى مكتبة التوابع ويمكن استدعائها عند الحاجة.

2-5- المكونات الأساسية لبيئة عمل c++.

Basics of a Typical C++ Environment

عند كتابة برنامج بلغة C++ وحتى يصبح قابل للتنفيذ لا بد من إجراء ما يلي:
الكتابة edit: يمكن أن يكتب بواسطة محرر نصوص عام ويجب أن يخزن بامتداد .c or .cpp or .cxx وفق برنامج المطابقة المستخدم، وعند استخدام محرر نصوص خاص لكتابة برامج بلغة C++ نجد أنه يلون الكلمات المفتاحيه بلون ازرق والتعليقات بلون أخضر وهكذا ويقوم بإسناد اللاحقة المناسبة بشكل افتراضي.

1- مرحلة ما قبل الترجمة preprocessor directives :عند طلب الترجمة يتم تنفيذ مرحلة ما قبل الترجمة التي تهدف إلى التعامل مع التوجيهات ما قبل الترجمة preprocessor directives حيث تحدد هذه التوجيهات جملة من المعالجات والإجراءات الواجب تنفيذها على نص البرنامج قبل عملية الترجمة، مثل احتواء لنصوص ملفات أخرى ضمن الملف المطلوب ترجمته بالإضافة إلى القيام بجملة من التعويضات لعبارات ضمن نص البرنامج.

2- الترجمة compile :يتم ترجمة الملف إلى لغة الآلة وهو ما يدعى object code ويتم تخزينه على القرص وهنا يتم اختبار الخطأ ضمن البرنامج syntax error

3- الوصل linking : تتضمن برامج C++ استدعاء لتوابع تم تعريفها في مكتبة ما أو في المكتبة المعيارية فيتم الوصل بين استدعاء التوابع والتوابع في مكان وجودها وعند نجاح المراحل السابقة نصل إلى ملف قابل للتنفيذ.

4- الشحن loading :قبل تنفيذ البرنامج يجب تحميله إلى الذاكرة إن لم يكن فيها.

5- التنفيذ executing : يبدأ الحاسب تنفيذ التعليمات.

2-6- بنية البرنامج بلغة c++

Program structure in C++

قد يبدأ البرنامج بالعبارة // the first program in C++ والتي تدل بأن هذا السطر هو تعليق Comment وتوضع التعليقات في البرنامج من أجل زيادة الإيضاح والتعليم لن يتم التدقيق فيه من قبل المطابق عند إنشاء البرنامج التنفيذي وبالتالي يمكن أن يتضمن كتابات بالطريقة التي تجدها أوضح وتوجد ثلاث طرق من التعليق:

- التعليق السطري والذي يبدأ من مكان وجود // وحتى نهاية السطر.
- التعليق لجزء من سطر أو أكثر من سطر: يبدأ مع ظهور /* وينتهي مع ظهور */ وقد يكون كلمة من سطر أو أكثر.

السطر #include<iostream> والذي يتضمن بين قوسيه أحد ملفات الرأس الضرورية لعمل التوابع المستخدمة ضمن البرنامج،

كل سطر يبدأ # يدعى بسطر توجيه لمرحلة ما قبل المترجم والتي يقوم بإخبار ما قبل الترجمة بضم الملف الرأسي iostream الذي يتضمن كافة التوابع التي تتعامل مع الإدخال القياسي من لوحة المفاتيح والإخراج على وحدة الخرج الرئيسة الشاشة.

بعد كتابة التوجيهات الضرورية لعمل البرنامج يمكن تعريف الثوابت والمتحولات العامة في البرنامج، كما يمكن الإعلان عن التوابع المصممة من قبل المستخدم (تعريفها) أو كتابتها بالكامل، وبعد ذلك يبدأ جسم البرنامج الرئيسي بالتابع main() حيث لا يوجد برنامج في لغة c++ بدون تابع رئيس اسمه main( ) يليه قوسان ويسبق بنوع المعطيات التي يعيدها وإذا لم يسبق بنوع معطيات فإن النوع هو int

وهذا السطر يدعى برأس التابع head ولكل تابع رأس وجسم body و يتوضع الجسم تحت الرأس ما بين القوس الكبير المفتوح نحو اليمين { والقوس الكبير المفتوح نحو اليسار } وكافة تعليمات التابع ضمنهما.

السطر التالي هو تعليمة خرج;" cout << " welcome to cpp

حيث سيتم دفع كل ما بين إشارتي الاقتباس(" ")quotation marks إلى مجرى الخرج القياسي standard output stream والمعبر عنه cout والمتصل بوحدة الخرج القياسية الشاشة ويمكن أن تكون وفق التالي:

1. في كل تعليمة خرج يجب أن تكرر العبارة السابقة
2. يمكن أن نضع في البرنامج using cout وبالتالي لا داعي لتكرارها.
3. ويمكن أن يلحق به h. بالمكتبة iostream في البداية لتصبح

#include<iosteam.h>

العملية >> ترمز للإدخال صمن المجرى ويتم من خلالها إدخال ما هو موجود على يمينها إلى مجرى الخرج

وكل عبارة يجب أن تنتهي بفاصلة منقوطة ";" semicolon. ماعدا سطر رأس التابع وتعليمة include و using وسطر define و { } بعد الاقواس الكبيرة السابقة والصغيرة ( ) .

البرنامج التالي يطبع العبارة welcome to cpp على الخرج القياسي الشاشة.

```
// the first program in C++
#include<iostream>
int main() {
cout<< " welcome to cpp";
return 0;}
```

وخرج البرنامج هو التالي:

```
welcome to cpp
```

وهنا يعيد قيمة الصفر إلى نظام التشغيل للحاسب ليخبره أن البرنامج نفذ بنجاح وتم الوصول إلى هذه النقطة التي تنهي البرنامج بإرادة المبرمج ويتم ذلك من خلال تعليمة return، ويمكن أن يكون التابع main () لا يعيد أي قيمة لذلك يسبق بـ void ولا يتضمن تعليمة return ضمنه، والتابع main() يبدأ منه تنفيذ البرنامج

ويمكن أن ينفذ البرنامج وفق التالي :

```
// the first program in C++
#include<iostream>

main()
 {
cout << " welcome\n to \ncpp";
return 0;
}
```

وخرج البرنامج هو التالي:

```
Welcome
to
cpp
```