الجمهورية اليمنية

جامعة إب

# مقدمة الى البرمجة بلغة C++

## Introduction to Programming using C++

دكتور:
نشوان المجمر

**2016**

# محاضرة (1)

## تقسيمات لغات البرمجة, مفاهيم البرمجه ومقدمة عن لغة C++

### 1. مقدمة عن لغات البرمجة ومفاهيم البرمجة

من أوائل اللغات التي ظهرت هي لغة Assembly وهي من أصعب لغات المستوى العالي ثم تلى ذلك ظهور لغات تستخدم مفردات اللغة الانجليزية مثل Fortran

تنقسم لغات البرمجة إلى قسمين :

1- لغات المستوى المنخفض Low Level Language مثل Assembly
2- لغات المستوى العالي High Level Language مثل Basic

أدى تطور لغات البرمجة إلى ظهور الكثير من المفاهيم منها:

A - البرمجة التركيبية Structured Programming

تعتمد البرمجة التركيبية على تقسيم البرنامج إلى أجزاء اومقاطع (دوال) ويعطى لكل جزء أسم خاص به ثم يقوم المبرمج بعد ذلك باستدعاء هذه الأجزاء التي يقوم كل منها بأداء مهمة محددة. يتم تقسيم البرنامج إلى دالة رئيسية ودوال فرعية

- Main function
- Other functions

فالبرنامج في البرمجة التركيبية: عبارة عن مجموعة من الدول كل منها يؤدي مهمة محددة وتستدعى هذه الدوال بعضها البعض.

مع تطور البرمجة وزيادة التعقيد في البرامج تطورت مفاهيم البرمجة فظهرت البرمجة بالأهداف OOP ( Object – Oriented Programming ) وتفيدنا البرمجة بالأهداف بأننا نستطيع تمثيل الأشياء المادية المحيطة بنا تمثيل حقيقي.

B- البرمجه بالأهداف Object – Oriented Programming (OOP)

تعتمد البرمجه بالأهداف علي تقسيم البرنامج إلى مجموعة من الفصائل( Classes ) فوحدة بناء البرنامج في OOP هي الفصيلة( Class )

فالبرنامج في OOP : عبارة عن مجموعة من الأهداف ( العناصر) Objects التي تتفاعل مع بعضها البعض. ولكي نستطيع أن ننشئ اياَ ًَ من

العناصر لابد أن بكون لكل عنصر قالب حتى نستطيع بناء عناصر متشابهة في كل الصفات والشكل. هذا القالب يسمى Class

نستطيع تمثيل الأشياء التي تحيط بناء باستخدام البرمجة بالأهداف ونستطيع أستخدمها من دون الحاجة إلى معرفة مما تتكون هذه الأشياء فنحن نهتم بالوظيفة التي يؤديها الـ Class دون أن نهتم بكيفية بناءة. بعد أنشاء العنصر Object نقوم بالتعامل مع الـ Class من خلال هذا العنصر

## 2. مقدمة عن C & C++

في عام 1960 صممت لغة CPL (Combined Programming Language ) وفي عام 1967 انبثقت لغة BCPL من لغة CPL على يد مارتن ريتشارد ثم قام ثومسون بتطوير لغة B وكل هذه اللغات تعتبر من لغات المستوى الأدنى ( Low Level Language ) أي القريب من لغة الآلة مثل لغة التجميع (Assembly ) في عام 1972 قام ريتشي باستنباط لغة جديدة من B أخذ منها أحسن تعليماتها وأضاف إليها أوامر جديدة و أنواع جديدة من البيانات وكثير من الدوال التي تفيد المبرمج وسميت هذه اللغة بلغة C ومنذ ذلك التاريخ أخذت لغة C شهرة واسعة لأنها تنتمي إلى لغات المستوى الأعلى من حيث سهوله الإستخدام ومن ناحية أخرى تنتمي إلى لغات المستوى الأدنى من حيث القدرة على مخاطبة مكونات الجهاز(Hardware) ومع نجاح وانتشار لغة C انتشرت لهجات كثيرة للمتحدثين بلغة C وكاد يحدث معها مع حدث للغة بيسك من وجود بيسك خاص لكل نوع من أنواع الأجهزة مثل (IBM-Apple ) و غيرها من الأجهزه, لولا أن قام معهد القياسات الأمريكية بعملية توحيد لهذه اللهجات فأصدر في عام 1983 اللغة القياسية أنسي سي ANSIC .

يوجد عدة مترجمات للغة C مثل (Borland c , Microsoft c ) تتوافق مع ANSI C

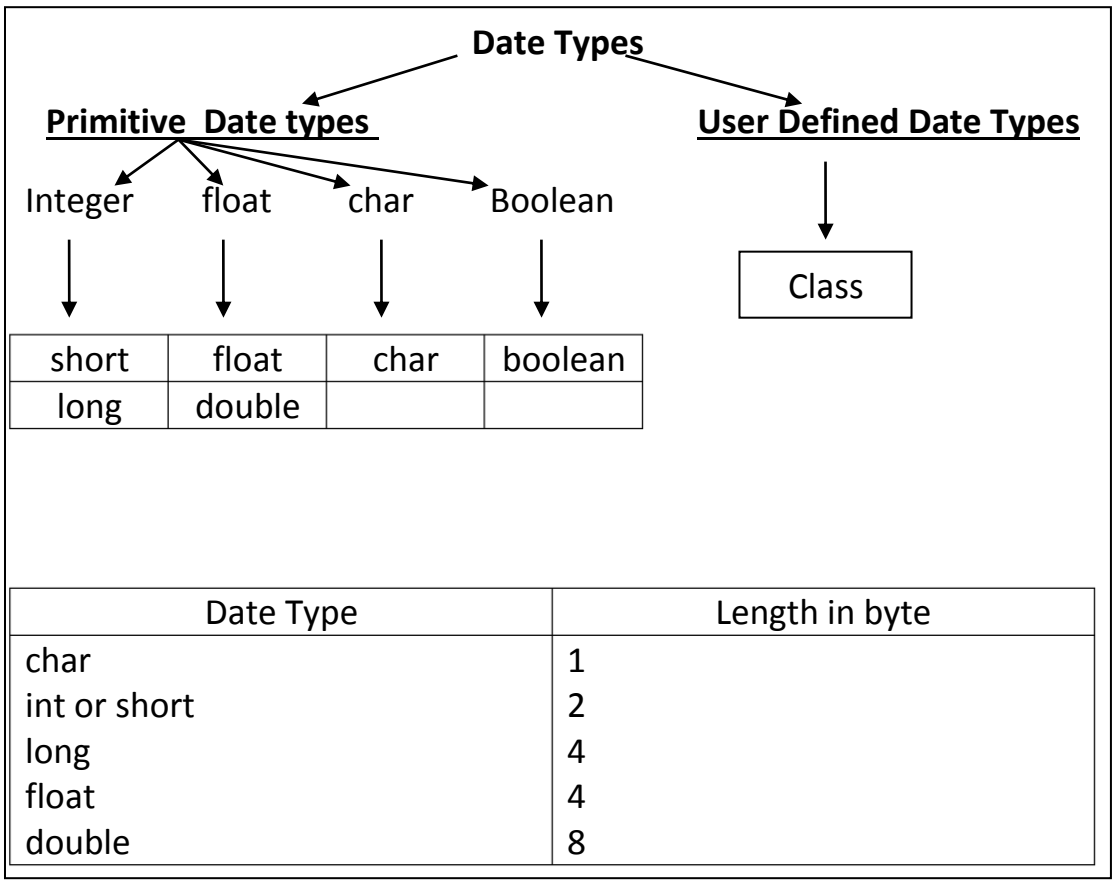
## 3. مميزات لغة C

1. لغة عامة : تصلح لعمل برامج قواعد البيانات والرسومات والحسابات ونظم التشغيل وغيرها.
2. لغة تركيبية ( Structured Language ): البرنامج المكتوب بلغة سي عبارة عن دالة رئيسية و مجموعة من الدوال الأخرى وكل داله من دوال البرنامج تنفذ مجموعة من الأوامر, بالإضافة إلى أن لغة (C++) المطورة من لغة C هي لغة برمجة بالأهداف ( OOP )وفي نفس الوقت لغة برمجة تركيبية (SL).
3. تتعامل مع البت: حيث تستطيع ان تقرا وتكتب وتغير وتقوم بعمليات على مستوى الـ Bit والبت هو أصغر وحدة لقياس المعلومات داخل الكمبيوتر وهي جزء من

ثمانية أجزاء (Byte ) وهذه الميزة جعلتها متخصصة في بعض مجالات التحكم الآلي والروبوت وبرامج معالجة الصور وضغط الملفات .

4. لغة متنقلة (Portal Language) فيمكن للبرنامج المكتوب بلغة الـ C أن تعمل مع أكثر من جهاز مثل IBM , Apple والأجهزة المتوسطة والكبيرة
5. لغة سريعة: وذلك بسب أن أدوات لغة C تتعامل مباشرة مع الآلة مما يختصر وقت التنفيذ.
6. لغة قياسية : معظم مترجمات اللغة تتوافق مع اللغة القياسية ANSIC
7. لغة نظام التشغيل UNIX: مما يدل على ثراء هذه اللغة.

## 4. Date Types In C++ أنواع البيانات




| Date Type | Length in byte |
|---|---|
| char | 1 |
| int or short | 2 |
| long | 4 |
| float | 4 |
| double | 8 |

## 5. المتغيرات Variables

المتغير عبارة عن عنوان لمكان في الذاكره نستطيع تخزين معلومة فيه أثناء عمل البرنامج و نستطيع تغيير القيمه المخزنه في هذا المتغير اثناء كتابه او نتفيذ البرنامج. حجم هذا المكان في الذاكره و القيمة المخزنة داخل هذا المكان تتوقف على نوع المتغير (Type ).

يتم الإعلان عن المتغير بكتابة النوع ( Type ) ثم أسم المتغير مع إمكانية إعطاء قيمة ابتدائية ويشترط في الاسم أن لا يبدأ برقم ولا يكون بين حروفه مسافة أو قوس أو أحد الحروف الخاصة مثل # أو @ و غيرها من الشروط التي سنتطرق لها لاحقاً, يمكننا مثلا الإعلان عن المتغيرات بالطريقه التاليه :

```
Int  a;
float b;
double number=3.6;
boolean male=false;
char ch;
```

### 6. الجمل والتعبيرات Statements & Expressions

الجمل statements : هي أمر بسيط مكتوب بلغة برمجة (مثلاً C++ ) يؤدي مهمة معينة

التعبير Expression : هي جملة تنتج قيمة ( مثلا تنفيذ عملية حسابية ما )

```
int  x=5;       // Statement
int y=1;        //Statement
int z=x+y;     // Expression
```

### 7. الثوابت والتعليقات Constants &Comments

الثوابت Constant : وهي قيمة لا تتغير أثناء عمل البرنامج ولكي نعرف ثابت في لغة C++ نضيف كلمة const بالصورة:

```
const double pi=3.141536
```

التعليقات Comments: وهو نص لا يلتفت إلية المبرمج أثناء عملية الترجمة compilation فالتعليقات مهمة للغاية لأننا في بعض الأحيان ننسى كيفية عمل جزء معين من البرنامج أو يريد الآخرين فهم عمل برنامج قمنا بكتابته.

<u>هناك نوعين من التعليقات:</u>

A. تعليق السطر الواحد, لكي نكتب تعليقا مكونا من سكر واحد يجب كتابه علامة // و بعدها نكتب نص التعليق
   مثلا // This is single line comments
   وهذا النوع خاص بالـ C++
B. تعليق مكون من عدة أسطر, لكي نكتب تعليقا مكونا من عدة اسطر لابد من حصر هذا التعليق بين الرمزين التاليين /* */ مثلا

```
/* This is multiple comment in my first  c++
                                    program*/
```

## 8. المؤثرات Operators

هي الرموز التي تربط بين المتغيرات والثوابت لإنشاء علاقة ما أو معادلة ما.

**أنواع المؤثرات:**

A. المؤثرات الحسابية Arithmetic Operators
(+،-،++،--، *، / ، %،...)

B. المؤثرات المنطقية Logical Operators
(&& , || ,  !،...)

C. المؤثرات العلائقيه Relational Operators
(==,<=  , < , > ، >=،!=،...)

مثال 1

| Example 1 | العملية | النتيجة |
|---|---|---|
| | 10>8 | 1 |
| | 10==8 | 0 |
| المؤثرات العلائقية | 10!=8 | 1 |
| | 10>=8 | 1 |

مثال 2

| Example 2 | العملية | النتيجة |
|---|---|---|
| | 10>8 && 9>7 | 1 (True) |
| المؤثرات المنطقية | 10<8  ||  9>7 | 1(True) |
| | !(10!=8) | 0 (false) |

مثال 3 بفرض ان قيمة a =6

| Example 2 | العملية | النتيجة |
|---|---|---|
| | a=a/3  أو  (a/=3) | 2 |
| المؤثرات الحسابية | a=a/3⇨a=2  a/=3⇨ a=2 | |
| | a=a+1  أو  (a++) | 7 |
| | a=a-1  أو  (a--) | 5 |

**9. أسبقية التعامل مع المؤثرات**

**A.** الحسابية

() ⇨ ++ ⇨ --⇨ * ⇨ / ⇨ % ⇨ + ⇨ -

**Example int answer**

**answer= 3+5*3-10%4**

**1.** 3 + 5 * 3 - 10 % 4

15

**2.** 3 + 5 * 3 - 10 % 4

2

**3.** 3 + 15 =18

**4.** 18 - 2 = 16

⇨ **The result is 16**

## B. المنطقية و العلائقيه

! → <,>,<=,>=,== → && → ||

**Example**

1. **(3>5) && (2==1) || (7==7)**

   **true**

2. **(3>5) && (2==1) || (7==7)**

   **false**

3. **(3>5) && (2==1) || (7==7)**

   **true**

4. **true && false**

   **false**

5. **false || true**

   **true**

   ⇨ **The result is true**

# محاضرة (2)

## بناء البرامج في لغة C++

تحدثنا في المحاضرة (1) عن لغات البرمجة ومفاهيمها وتقسيمات لغات البرمجة إلى لغات ذات مستوى منخفض ولغات ذات مستوى عالي وتحدثنا عن مفاهيم البرمجة ( البرمجة التركيبية والبرمجة بالأهداف ) ، أعطينا نبذة عن تأريخ لغة C/C++ ومزايا هذه اللغة ، أعطينا مقدمة عن نوع البيانات وعن المتغيرات والجمل و التعابير والثوابت والتعليقات وتحدثنا عن المؤثرات وأنواعها ( العلائقية ، الحسابية ، المنطقية)

### 1- بعض المسلمات والتعريفات

**بعض التعريفات :**

البرمجة : هي أعطاء أوامر للحاسوب لتنفيذ مهام ما أو لحل مشكلة ما. لغة هذه الأوامر هي لغة يفهمها الحاسوب (اي كتابة برنامج)

البرنامج : البرنامج هو مجموعة أوامر أو جمل Statements مكتوبة بلغة يفهمها الحاسوب للوصول إلى الهدف الذي من أجلة كتب البرنامج .

**بعض المسلمات :**

1. البرمجة: هي حلول منطقية لمسائل معينة ، فمثلاًً عندما يريد مدير الموظفين أن يعرف عدد الموظفين الذين مرتباتهم أكثر 60 ألف ريال فإن كل ما عليه هو أن يمر على الموظفين و أن يرى كشف مرتبات الموظفين ، الطريقة المنطقية تقول بأن مدير الموظفين سيسمي متغيراً باسم ( عدد الموظفين الذين رواتبهم أكثر من 60 ألف ريال ) أو ( اختصاراً نسميه عدداً) ويعطى القيمة صفر 0 قبل أن يبدأ في مراجعة كشف المرتبات وعند مراجعة الكشف ، عندما يرى أن أحد الموظفين راتبه أكثر من 60 ألف ريال يبدأ في زيادة المتغير عدد بمقدار 1 وهكذا حتى ينتهي من مراجعة الكشف بأكمله.

عدد= 0

نبدأ بمراجعة الكشف

كلما رأينا في الكشف مرتب أكثر من ستين ألف ريال نفذنا العملية: عدد=عدد+1

وهكذا حتي ننتهي من الكشف

2. دائما هناك طرق مختلفة لحل مسألة ما فكثير من المسائل يمكن حلها بأكثر من طريقة لكن غالباً هناك طريقة أفضل من طرق أخرى فالمهم الوصول إلى حل وإذا كان هناك متسع من الوقت فأننا نبحث عن الطريق الأفضل للحل.
3. من العادات الحسنة في البرمجة والمهمة أن يتمك تجزيء البرنامج إلى أجزاء أو إجراءات بحيث يكون كتابة كل جزء أسهل بكثير من كتابة البرنامج مره واحدة وكذلك في تصحيح الأخطاء وعند البحث عنها في جزء ما بعد كتابة الأجزاء يتم ربطها مع بعضها البعض لتكوني البرنامج .
4. خطوات 1: البرنامج ( المطلوب ، المدخلات ، المخرجات) لابد أن نفهم المطلوب والمدخلات والمخرجات .

الخطوة 1 : تسمى التحليل Analysis أو تعريف المشكلة problem definition فلا يمكن أن نكتب برنامج لحل مشكلة ما إذا كنا لم نفهم المطلوب من المسألة جيداً

الخطوة 2:نقوم بتصميم الحل أي البرنامج، بمعنى وضع الخوارزمية ( الطريقة التي سوف توصلنا إلى الحل )

الخطوة 3 : تحويل الخوارزمية إلى لغة يفهمها الحاسوب ( تحويل الخوارزمية على برنامج مكتوب يفهما الحاسوب كلغة C++ )

الخطوة 4 : اختيار البرنامج للتأكد من خلوة من الأخطاء وانه يقوم بحل المسألة المطلوبة

الخطوة الأخيرة: تنفيذ البرنامج للحصول على النتائج.

## 2- C++ language & Program structure in C++

حروف لغة C++ هي التالية:

1- الحروف الهجائية الإنجليزية الكبيرة والصغيرة a,b,c….x,y,A,B,C
2- الأرقام العربية ….10…….5 ,4 ,3 ,2 ,1 ,0
3- الرموز الخاصة #,%,&,!
4- المعاملات مثل =>,<<,>>,/,*,-,+

المعرف identifier : هو ذلك الاسم الذي يخزن فيه قيم مثل الثابت أو المتغير أوأن يكون اسما لدالة ما وشروطه هي:

- أن يتكون من حرف أو أحرف أو أحرف وأرقام و قد تكون هذه الأحرف كبيره أو صغيره و بشرط أن يبدأ بحرف أو شرطه تحتيه under score و ليس برقم
- خالي من أي رمز خاص ماعدا الشرطه التحتيه under score
- يفضل أن يكون له معنى واضح وذو مدلول
- أن لا يكون من الكلمات المحجوزة في اللغة مثل:

include, cin, cout, int, double, void, main, define, for, while, do, …..

## بنية البرنامج في لغة C++

--- Documentation section (تعليقات) معلومات عن البرنامج

For Example: /* */

--- link Section إعطاء الأمر للمترجم بتضمين بعض الدوال

For Example: # include <iostream>

--- Definition Section بعض التعريفات

For Example: const double PI=3.14;

--- Global declaration section بعض المتغيرات العامة التي تستخدمها كل الدوال

For Example: int sum;

--- main function section الداله الرئيسيه

```
{
    Declaration section
    Executable section
}
```

/* Subprogram section or other functions section*/ بقيه دوال البرنامج

```
function 1
{
  statements
}
.....
function N
```

```
{
statements
}
```

## البرنامج الأول في C++

```
// This is my first program in c++
#include<iostream.h>
int main()
{
cout<<"welcome to c++";
return 0;
}
```

#include<iostream.h> هذا السطر يتم فيه تضمين مكتبة الإدخال والإخراج في لغة C++

وذلك لأن أساليب الإخراج والإدخال غير معرفه في اللغة ولكنها موجودة في المكتبات الخاصة باللغة ويتم التضمين باستخدام الأمر include , يسمى أي ملف ينتهي بالامتداد h بالملف الراسي Header file ,تحتوي الملفات الرأسيه header files على فئة وتراكيب بيانات ودوال وثوابت ويتم أنشاء هذه الملفات عندما تكون هذه العناصر البرمجية عامة الاستخدام أي أنها تستخدم في عدة برامج ومن ثم بدل كتابتها كل مرة يتم كتابتها مرة واحدة في ملف راسي ومن ثم يتم تضمينها في البرامج التي نحتاج فيها هذه الملفات.

# هذا الرمز يسبق عملية التضمين include وكل أمر يسبق بهذا الرمز فانه يسمى موجه ما قبل المعالجة preprocessor directive أي أن مترجم اللغة يقوم بتنفيذ ما يمليه هذا الموجة قبل أن يترجم البرنامج.

// كل ما بعد هذا الرمز يعتبر تعليقا و لا يلتفت اليه المترجم عند تنفيذ البرنامج, فلكي يكون البرنامج مبرمجاً بطريقة جيدة لا بد أن يحتوي على تعليقات لتسهيل عملية تطوير البرنامج وسهولة فهمة من قبل الآخرين.

الدالة Main وتتبع بقوس {بداية الدالة وقوس نهاية الدالة } ,من هذه الدالة يتم تنفيذ أي برنامج بالـــC++ فلابد من وجودها في اي برنامج ليتم تنفيذه ويتم تنفيذ الجمل البرمجية المحتواة داخلها جمله بعد أخري.

cout – هي اختصار للجملة course output أي منهج الخرج والذي هو الشاشة وهو كائن يقوم بإخراج ما يأتي بعدة على الشاشة (مجرى الإخراج) وهذا الكائن معرف في المكتبة iostream لذلك يتم تضمينها مسبقاً

>> معامل الإخراج والذي يقوم بإرسال ما يأتي بعدة إلي الكائن cout كما في الامثله التاليه:

```
int Data=10;

cout<<Data;

cout<<"welcome to c++";
```

كل جملة في c++ تنتهي بـــ ; (فاصلة منقوطة)

Return 0; :تقوم بإنهاء البرنامج والرقم 0 يعنى إنتهاء البرنامج بنجاح

}نهاية جسم الدالة الرئيسية

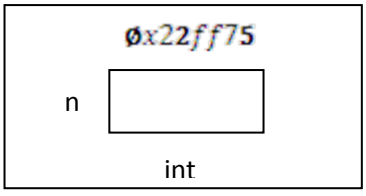
### 3- المتغيرات وأنواع البيانات وحجم أنواع المتغيرات في الذاكره وبرنامج توضيحي

كما أسلفنا بأن المتغير هو أسم لموقع في الذاكرة العشوائية RAM هذه المواقع يتم تخزين البيانات فيها آثناء عمل البرنامج حسب نوع المتغير أي أن نوع و حجم القيمة المخزنة في هذه المواقع من الذاكرة متوقف على نوع المتغير وهذا المتغير نستطيع تغيير قيمته أثناء عمل البرنامج

كل خلية من الذاكرة يعطيها نظام التشغيل عنوانا في هيئة النظام السادس عشر, فمن المستحيل إذا أردت استخدام هذه الخلايا لتخزين البيانات فيها أن تقوم بحفظ عناوينها ولذا يتم استخدام المتغيرات لإعطاء الخلايا أسماء واضحة تسهل علينا التعامل مع الذاكرة كما أن المتغير يمكن أن يأخذ أكثر من خلية ذاكرة إذا لم تكن الخلية الواحدة كافية لحفظ قيمة المتغير

For example:

```
int  n=5;
```



برنامج يوضح حجم متغيرات مختلفه في الذاكره بالبايت:

```
#include<iosteam.h>
\\ data types
main()
{
char c='5';
int i=-92,k;
short si=58;
long li=-999;
unsigned char us='$';
unsigned short us;
unsigned long ul;
float f;
double d;
long double ld;
// بطريقتين sizeof(variableName) نستخدم الداله
k=sizeof(c);  // الطريقة الأولى  بحفظ ناتج الداله في متغير ما و من ثم اخراج قيمه هذا المتغير
cout<<"\n"<<"the size of char is:"<<k<<"byte";
// الطريقة الثانية بإخراج قيمه الداله مباشره
cout<<"\n the size of an int  is:"<<sizeof(i)<<"byte\n";
```

```
cout<<"\n the size of  short  is:"<<sizeof(si)<<"byte\n";

cout<<"\n the size of  long int  is:"<<sizeof(li)<<"byte\n";

cout<<"\n the size of  unsigned char  is:"<<sizeof(us)<<"byte\n";

cout<<"\n the size of  unsigned short   is:"<<sizeof(us)<<"byte\n";

cout<<"\n the size of  unsigned long   is:"<<sizeof(ul)<<"byte\n";

cout<<"\n the size of a float    is:"<<sizeof(f)<<"byte\n";

cout<<"\n the size of a double    is:"<<sizeof(d)<<"byte\n";

cout<<"\n the size of a long double    is:"<<sizeof(ld)<<"byte\n";

return 0;

}
```

**مخرجات البرنامج:**

```
the size of char is: 1 byte

the size of an int is : 2 bytes

the size of  short  is: 2 bytes

the size of long int is: 4 bytes

the size of  unsigned char is: 1 byte

the size of  unsigned short is: 2 bytes

the size of  unsigned long is :4 bytes

the size of  a float is :4 bytes

the size of  a double is :8 bytes

the size of   long double is :10 bytes
```

### 4- كتابة برنامج حسابي بسيط بلغة++C

البرنامج التالي يقوم بحساب عمر شخص بالايام و الساعات والثواني

```
#include<iostream.h>
main()
{
// declaration section
short age_in_years=80
int    age_in_days;
int    age_in_hours;
long int age_in _seconds;
string name="Ahmed"
/* print person and his age*/
cout<<"\n Welcome Mr."<<name<<”!”<<endl;
age_in_days= age_in_years*365;
cout<<"\n your age calculated in days:"<< age-in-days;
cout<<"\n your age calculated in  hours :"<< age_in_days*24<<endl;
age_in_second=age_in_hours*3600;
cout<<"\n your age calculated in seconds:"<< age-in-second<<endl;
}
```

# محاضرة(3)

## أدوات الإدخال والإخراج input and output tools

**1- syntax (نحو اللغة)**

تعريف ال syntax: هو بناء الجملة وترتيب كلماتها في إشكالها وعلاقاتها الصحيحة.

1. الإعلان عن متغير :

- Type variableName;
- Type variableName=VariableValue;
- Type variableName1, Type variableName2, ...,Type variableNameN;
- Type variableName1=value1,variable2,…,variableN;

2. الثوابت Constants :

- cons constantName = constantValue;

3. جمل الإدخال : (cin )

- cin>>variableName;

عندما يصل البرنامج إلى جملة cin يتوقف منتظراً إدخال قيمة للمتغير عن طريق لوحة المفاتيح ثم تخزن تلك القيمة في عنوان المتغير المخصص له بالذاكرة.

- cin>>variableName1>> variableName2>>…>> variableNameN;

4. جمل الإخراج cout:

```
cout<<argument;
cout<<"format"<<argument;
cout<<["format"<<]argument1<<argument 2<<…<< argument n;
```

5. التوجيهات

```
#include<filename.h>
```

6. الدالة الرئيسية ()main تمتلك إحدى الصيغ العامه التاليه:

```
1.
void main()
{
Any statements or expressions
}
```

2.

```
int main()
{
Any statements or expressions
return 0;
}
```

3.

```
main()
{
Any statements or expressions
}
```

**2.برنامج لإدخال أنواع مختلفة من البيانات وإخراج معلومات عن حجم هذه البيانات في الذاكرة**

```
// Date type and the size of types in memory
#include<iostream.h>
int main()
{
char c;
short int si;
long int li;
float f;
double d;
cout<<"\n input a char ";
cin>>c;
cout<<"\n input a short integer number ";
cin>>si;
cout<<"\n input a long integer number ";
cin>>li;
cout<<"\n input a float number ";
```

```
cin>>f;
cout<<"\n input a double number";
cin>>d;
// output the results
cout<<"the size of c is : "<<sizeof(c)<<endl;
cout<<"the size of si is : "<<sizeof(si)<<endl;
cout<<"the size of li is : "<<sizeof(li)<<endl;
cout<<"the size of f is : "<<sizeof(f)<<endl;
cout<<"the size of d is : "<<sizeof(d)<<endl;
return 0;
}
```

لتنظيم عملية إخراج البيانات في دالة الإخراج تستعمل حروف الهروب Escape character وتقوم بوظائف معينة عند أخراج البيانات على الشاشة ، هذا الحروف تكون مكتوبة بين علامتي تنصيص.

بعض هذه الحروف :

\n سطر جديد
\b مسافة إلى الخلف
\t مسافة تاب أفقية
\r الرجوع إلى بدية السطر
\a الإنذار بالجرس
\' لطباعة علامة
\" لطباعة علامة
\? لطباعة علامة الاستفهام

**3. نتيجة عمل المؤثرات العلائقية والمنطقية وإخراج هذه النتائج**

- المؤثرات العلائقية

```
==
!=
>
<
<>
>=
```

<=

المؤثرات العلائقية عادة تكون بداخل قوسين ونتيجتها اما 0 أو 1

مثال1

```
cout<<(6>2);
```

النتيجة هنا هي 1

مثال2

```
cout <<(3>=6);
```

النتيجة هنا هي 0

المؤثرات المنطقية هي مؤثرات تقوم بالربط بين تعبيرات او عناصر منطقية لتجعلها تعبيراً واحداً (النتيجة أما true or false)

مثال1

```
cout<<(20==20&&678);
```

النتيجة هنا هي0

مثال2

```
cout<<(!0) ;
```

النتيجة هنا هي 1

**4. برنامج توضيحي للعمليات الحسابية وعمليات التخصيص**

| () ⇨ ++ ⇨ --⇨ * ⇨ / ⇨ % ⇨ + ⇨ - |
|---|

```
int main ()
{
short  a,b,c,d,e;
d=e=30;
a=4;
b=--a+1;
c=++a+b++;
cout<<"A="<<a<<"\t"<<"B="<<b<< "\t"<<"c="<<c<<"\n";
c+=--a+--b;
a==b=c-(a*b);
d/=a+b;
e=e/a+b;
```

```
cout<<"D="<<d<<"\t"<<"E="<<e;
return 0;
}
```

| C | B | a | e | d | رقم العمليه |
|---|---|---|---|---|---|
| | | | 30 | 30 | 1 |
| | | 4 | | | 2 |
| | 4 | 3 | | | 3 |
| 8 | 5 | 4 | | | 4 |
| 15 | 4 | 3 | | | 5 |
| | 3 | 3 | | | 6 |
| | | | | 5 | 7 |
| | | | 13 | | 8 |

**الإخراج على الشاشة:**

A=4 B=5 C=8 D=5 E=13

**5. برنامج لحساب وطباعة مساحة ومحيط دائرة نصف قطرها (r)**

المطلوب: حساب مساحة محيط دائرة

المعطيات: نصف قطر الدائرة r=6.5 حيث أن المساحة = $PI*r^2$ ومحيط الدائرة يساوي 2*PI*r .

```
#include<iostream.h>
int main ()
{
const double PI=3.1415926;
float r;
double area, circumference;
cout<<"input the radius of circle"<<endl;
cin>>r;
area=PI*r*r;
circumference=2*PI*r;
cout<<"\n the area of circle ="<<area<<endl;
cout<<" the circumference of circle ="<< circumference<endl;
return 0  ;
}
```

**الإخراج على الشاشة:**

input the radius of circle please

```
6.5
the area of circle=132.73228735
the circumference of circle=40.8407038
```

# محاضرة(4)

# Conditional Structure (if, if….else, switch)

# التراكيب الشرطية او جمل التحكم (إذا, إذا وإلا, انتقاء)

في البرامج السابقة نفذت العمليات بطريقة متسلسلة بمعنى تعليمة بعد أخرى بطريقة متتابعة.

بواسطة التراكيب الشرطية نتمكن من التحكم بتنفيذ خطوات البرنامج .

## 1- التركيب الشرطي البسيط ( if )

تقوم if بتحقيق أو تنفيذ عمليات ما إذا كان الشرط صحيحاً فقط.

الصيغة العامة:

- if (condition) statement;

  أو
- if(condition)
  {
  block of statement
  }

**مثال 1**

البرنامج يطلب من المستخدم ادخال عدد صحيح ثم يقوم باختبارهذا العدد, هل هو أكبر من الصفر؟

ثم يعطي رساله للمستخدم بان العدد موجب في حال كان العدد أكبر من الصفر.

```
#include<iostream.h>
int main()
{
int x;
cout<<"input the integer number :\n";
cin>>x;
if (x>0)
 cout<<"the number is positive";
```

```
return 0;
}
```

**مثال 2**

البرنامج يطلب من المستخدم ادخال عدد صحيح ثم يقوم باختبارهذا العدد, هل هو أكبرأم أصغر من الصفر؟ ثم يعطي رساله للمستخدم بأن العدد موجب أوبأن العدد سالب.

```
#include<iostream.h>
int main ()
{
int x;
cout<<" input the integer number\n";
cin>>x;
if(x>0)
  cout<<"the number is positive";
if(x<0)
  cout<<"the number is Negative";
return 0;
}
```

**مثال 3**

البرنامج يقوم بقراءة عددين حقيقيين ثم يرتب هذين العددين تصاعدياً

```
#include <iostream.h>
void main()
{
float a,b,temp;
cout<<"input the number 1 and number 2:"<<endl;
cin>>a>>b;
if(a>b)
 {
   temp=a;
```

```
        a=b;
        b=temp;
      }
     cout<<a<<"  "<<b;
     }
```

### 2- التركيب الشرطي الكامل ( if....else ) إذا ....وإلا

نستخدم هذه التركيب الشرطي لاختيار شرط معين, فإذا كان صحيح ينفذه ما بعد if وإلا ينفذ ما بعد else

الصيغة العامة:

- 
```
if (condition)
  statment1;
else
  statement 2;
```
أو
- 
```
if (condition)
{
  Block of statements
}
else
{
  Block of statements
}
```

مثال

```
#include<iostream.h>
int main()
{
int x;
cout<<"input the integer number \n";
cin>>x;
if(x%2==0)
cout<<"the number is even\n";
else
```

```
cout<<"the number  is odd \n ";
return 0;
}
```

## 3- تراكيب المؤثر الشرطي (Conditional operator ?)

**الصيغة العامة:**

```
variableName = (condition)?  statement1 :   statement2;
```

عمل المؤثر الشرطي(؟) مشابه لعمل التركيب الشرطي if..else

**مثال1**

أكتب برنامج لحساب وطباعة قيمة y إذا علمت أن

$$y = \begin{cases} 5 - x^2, & \text{if } x >= 0 \\ 2x^3, & \text{if } x < 0 \end{cases}$$

```
#include<iostream.h>
int main()
{
float x,y ;
cout<<"input c:";
cin>>x;
y=(x>=0)?  5-x*x : 2*x*x*x;
cout<<"Y=  "<<y;
return  0;
}
```

## 4- التركيب الانتقائي Switch

التراكيب الشرطية السابقة تتم فيها المقارنة بين قيمتين حيث تكون النتيجة أما صائبه أو خاطئة ولكن في بعض الأحيان علينا أن نقارن بين عدد من الحالات تبعاً لشروط مختلفة في هذه الحالة

نستطيع استخدام التركيب (switch ) والذي يقوم باختيار القيمة الصحيحة من عدد من القيم وحسب الشرط الموجود في الصيغة

الصيغة العامة:

```
switch (variableName)
{
case value _1: statement_1;
                break;
case value_2:  statement_2;
                break;
.
.
case value_N: statement_N;
                break;
default :default  statement;
}
```

variableName: هو ذلك المتغير الذي يتم اختياره ويكون من النوع صحيح(int) أو الحرفي (char)

case : تمثل نوع الحالة المناسبة بعد احتساب المتغير

value: تمثل قيمة المتغير ويمكن أن تكون عدداً موجباً او سالباً من النوع الصحيح(int) أو أن تكون حرفا (char)

break: وهي عبارة توقف وتستعمل عند أخر كل مجموعة جمل من جمل الحالة (case ) لتفادي الاستمرار في بقية الحالات وإذا لم تستعمل بعد أي حالة فان التعبير ينتقل إلى الحالة التالية لهذه الحالة

default : وتعني حالة إسقاط وهي اختيارية (يمكن عدم ذكرها في البرنامج ) وتنفذ عندما لا تتوافق قيمة المتغير مع أي قيمةvalue1 , value2, …, valueN

**مثال**

برنامج العمليات الحسابيه (+ ,- , *, □ )

```
#include<iostream.h>

void main()
```

```
{
float a,b;
char op;
cout<<"Enter two numbers:"<<endl;
cin>>a>>b;
cout<<"Enter operator :"<<endl;
cin>>op;
switch (op)
 {
        case'+':  cout<<a<<"+"<<b<<"="<<a+b;
                        break;
        case'-': cout<<a<<"-"<<b<<"="<<a-b;
                        break;
        case'*': cout<<a<<"*"<<b<<"="<<a*b;
                        break;
        case'/': if(b==0)
                        cout<<"error divide by zero";
                else
                        cout<<a<<"/"<<b<<"="<<a/b;
                break;
        default: cout<<"Error input operator ";
 }
```

}

# محاضرة (5)

# Loops( for, while ,do-while)

## الدورات أو الحـــلقـــات أو جمـــل التـــــكرار

**1- جملة التكرار for**
**2- جملة التكرار while**
**3- جملة التكرار do- while**

تعريف الحلقة **loop** : هي مجموعة من الجمل التي تستعمل لتكرار تنفيذ الأوامر أكثر من مرة وهي مهمة جدا لكتابه البرامج ً وهذه الجمل هي :

1- جملة التكرار for
2- جملة التكرار while
3- جملة التكرار do- while

### 1- جملة التكرار for

### 1-1 الصيغة العامة:

تستخدم for لتكرار تنفيذ عملية أو عمليات عدد محدد من المرات وتأخذ الصيغة العامة ل for الشكل التالي:

- for( variableName=init_value; condition; increment)
  statements;
  أو
- for( variableName=init_value; condition; increment)
  {
  Statement_1;

....
....
Statement_N;
}

إذا كنا نريد تكرار عدة جمل فلابد من كتابتها بين القوسين {}

**شرح الصيغه العامه:**

حيث أن:

init-value : هي القيمة الابتدائية

condition : هو شرط التكرار

increment : هي قيمه أو جمله الزياده

for(..) :الجمله for تعني اعطي لمعامل الزياده القيمة الابتدائية (init-value) و طالما ان الشرط (condition) صائب نفذ الجمل الموجوده في جسم الحلقه forً ثم قم بزياده أو نقصان قيمه معامل الزياده أي نفذ الجمله increment

**مثال1**

طباعة العدد 7 عشر مرات في أسطر مختلفة

```
/* for loop*/
#include<iostream.h>
main()
{
   int I;
   for (i=0;i<10;i++)
       cout<<"7"<<endl;
}
```

**مثال 2**

طباعة الأعداد من 1-10

```
#include<iostream.h>
main()
{
   int j;
   for(j=1;j<=10;j++)
```

```
        cout<<"j="<<j;
}
```

**مثال 3**

تكرار أكثر من جملة لعدد محدد من المرات باستخدام for في هذا المثال نقوم بحساب مجموع الأعداد من 1- 10

```
#include<iostream.h>
main()
{
        int   i,sum;
        sum=0;
        for(i=1;i<=10;i++)
        {
                sum=sum+i;
                cout<<"Now, the sum is:  " <<sum<<endl;
        }
}
```

### 2-1 تغيير مقدار الزيادة في الحلقة for

في بعض الأحيان نحتاج إلي أن يكون مقدار الزيادة أكثر من واحد حينئذ يلزم تغيير هذا التعبير إلى الصورة المطلوبة مثلا اذا كنا نريد أن يكون مقدار الزياده (2)

**i=i+2 ⟺ i+=2**

**مثال**

حساب مجموع الأعداد الزوجية من 1..10

```
#include<iostream.h>
main()
{
  int i , sum=0;
  for(i=2; i<=10; i+=2)
        sum=sum+I;    // or sum+=I;
}
```

### 3-1 تغيير معدل الزيادة بالسالب

أحياناً نحتاج إلى أن يكون معدل الزيادة بالسالب لحل بعض المسائل

**مثال:**

طباعة الأعداد من 10 إلى 1 ( من عشرة إلى واحد)

نستطيع حل هذه المسألة باستخدام for مع معدل زيادة بالسالب بالشكل التالي :

```
#include<iostream.h>
main()
{
   int i;
  for(i=10; i>0;i--)
     cout<<"I  is: "<<i<<endl;
}
```

### 4-1 الدوارة اللانهائية باستخدام for

ومعناها تكرار تنفيذ الجملة بدون شرط ولا يتوقف تكرار الحلقة أو الدوارة حتى يضغط المستخدم على المفتاحين (ctrl+break ) وشكل هذه الحلقة اللانهائية هو:

```
for( ;  ; )
```

لو كتبت الدوارة بالصورة التاليه ;( ;  ; )for ونفذ البرنامج فأن الجهاز يدخل في مسار لا نهائي ولا نستطيع حل هذه المشكلة إلا بإعادة تشغيل الجهاز .

# محاضرة(6 )
# بقية الحلقات (while & do...while )

## 2- الحلقة ( while )

تستخدم الحلقة أو الدوارة (while ) لتكرار تنفيذ جملة أو مجموعة من الجمل عدد غير معلوم من المرات يتوقف هذا العدد على شرط موجود بداخل الحلقة (while)

### 2-1 الصيغة العامة لـ (while ):

```
while (condition)
{
  statements
}
```

ومعناها طالما أن الشرط (condition) صحيح نفذ التعليمه أو التعليمات (statements ) وهي تقوم بتكرار الجملة أو مجموعة الجمل الموجوده في جسم الحلقه طالما كان الشرط صحيح وعندما يصبح شرط التكرار غير صحيح يتوقف تنفيذ الحلقة (while)

**مثال 1:**

طباعة الأعداد من واحد على عشرة

```
#include<iostream,h>
main()
{
   int j=1;
   while(j<=10)
   {
         cout<<"j="<<j;
```

```
                j++;
            }
        }
```

**مثال 2:**

أجمع الأعداد التي تبدأ بواحد وتزداد بمقدار واحد إلى أن يصبح المجموع اقل من (100)

```
#include<iostream.h>
main()
{
    int i=1;
    int sum=0;
    while(sum<100)
    {
        sum=sum+i;
        i++;
    }
    cout<<"the sum is :"<<sum;
}
```

### 2-2 الفرق بين For& while

الدوارة for دوارة أو حلقة تكرار عددية حيث تعتمد على العداد وينتهي التكرار فيها بانتهاء عدد مرات التكرار ، أما حلقة التكرار (While ) فهي شرطية أي تعتمد على الشرط الذي يلي الأمر (While ) حيث تتكرر الجمل التي تليه طالما كان الشرط صحيحاً وتنتهي الحلقة بكسر هذا الشرط. بالتالي فأن الاستخدام الأمثل للحلقة for هو تكرار عملية ما أكثر مرة بشرط أن يكون عدد مرات التكرار معلوم مسبقا والاستخدام الأمثل للحلقة(While ) هو التكرار بناء على شرط معين بغض النظر عن عدد مرات التكرار .

### 3-2 الحلقة اللانهائية باستخدام (While )

الحلقة اللانهائية مع for تأخذ الصورة for( ; ; )

الحلقة اللانهائية مع (While ) تأخذ الصورة while (1)

**مثال على الحلقة اللانهائية:**

```
#include<iostream.h>
main()
{
    int i=0;
    while(1)
     {
       cout<<"i is " <<I ;
      i++;
      }
}
```

يقوم هذا البرنامج بطباعة أعداد .....1,2,3,4, وهكذا حتى يضغط المستخدم على ctrl+c

**3- جملة التكرار do....while**

تستخدم الجملة ( do....while ) لتكرار تنفيذ جملة أو مجموعة من الجمل اكثر من مرة بناءً على شرط معين كما هو الحال مع ( while ) ولكن الفرق بينهما أن( While) تختبر الشرط أولاً فإذا كان صحيحاً تنفذ الجمل التالية لها وإلا فلا. أما جملة (do....while) فتنفذ الجمل التالية لها أولاً مره واحده ثم تختبر الشرط . فإذا كان صحيحاً تعيد التنفيذ وإلا توقف التكرار.

**الصيغه العامه:**

```
do
```

```
{
  statements
} while(condition);
```

ومعناها نفذ (do) الجمل ( statements ) طالما ( while) أن الشرط( condition) صحيحاً

**مثال 1:** أكتب برنامجاً لإدخال وطباعة عدد صحيح أوقف البرنامج عند إدخال القيمة 0 .

```
#include<iostream.h>
int main()
{
        unsigned long n;
        do
        {
                cout<<"Enter   number  (0 to end)";
                cin>>n;
                cout<<"you  entered :"<<n<<endl;
        } while(n!=0);
return  0;
}
```

**مثال2:** أكتب برنامج لحساب و طباعة مجموع الأعداد التالية: (3.0,3.5,4.0,4.5,…10.0)

الحل : نبدأ الجمع من العدد (3.0 ) إلى العدد (10.0) ومقدار الزيادة في كل مرة (0.5 )

**البرنامج :**

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

```
    float a=3.0,   sum=0.0;
    do
    {
        sum=sum+a;
        a=a+0.5;
    } while(a<=10);
   cout<<"the result is :" <<sum;
   return 0;
} // end of main
```

ملاحظه التعليمات التالية :

```
sum=sum+a;
a=a+0.5;
```

يمكن كتابتها بالصورة التالية:

```
sum+=a;
a+=0.5;
```

# محاضرة(7)

## جمل التكرار المتداخلة Nested loops

الدورات المتداخلة عبارة عن دوارة كبيرة (خارجية) تشتمل بداخلها على دوارة (داخليه) أو أكثر. بمعنى أن مجموعة التعليمات الموجودة بالدوارة الداخلية يتكرر تنفيذها طالما ان شرط التكرارصحيح بالنسبه للحلقه الداخليه فإذا انتهي تكرار الحلقه الداخليه, ينتقل التنفيذ إلى الدوارة الخارجية و يتم تكرار تنفيذها طالما ان شرط التكرار صحيح بالنسبه للحلقه الخارجيه, بمعني ان كل الدوارات الداخليه يتم تكرارها في كل مره يتم فيها تكرار الحلقه الخارجية.

### 1- جمل for المتداخلة

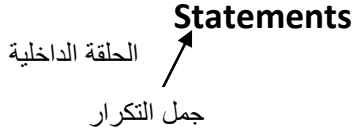
**تأخذ حلقات التكرار المتداخلة for الشكل العام التالي :**

```
For ( ……)
  For (……..)
    For (……)
      …….
        Statements
      ……
```

**فلو اخذنا حالة حلقتين متداخلتين فإنهما تكتبان بالطريقة التالية:**

```
For (……)
    For (……)
```

الحلقة الخارجية




**مثال1** : طباعة جدول الضرب من واحد إلى عشرة بالشكل التالي :

1*1=1 1*2=2 . . . 1*10=10

2*1=2 2*2=4 . . . 2*10=20

.

.

.

10*1=10 10*2=20 . . . 10*10=100

البرنامج :

```
#include<iostream.h>
main()
{
        int  i, j;
        for(i=1;i<11;i++)       // for 1
        {
                for(j=1;j<11;j++)    //  for 2
                {
```

```
                cout<<i<<"*"<<j<<"="<<i*j;
                cout<<"       ";
            } // end for 2
            cout<<"\n";
        } // end for 1
} // end of main
```

**مثال 2**: تتبع مخرجات البرنامج التالي:

```
# include < iostream .h >
Main( )
{
   Int  I, j ;
 For  ( i= 1;i<=3 ;i++)
   For ( j=1 ; j<=4 ; j++)
     Cout <<  i*j<< "        " <<endl ;
return(0) ;
}
```

الحلقة الداخلية تتكرر اربع مرات لكل مرة من مرات التكرار للحلقة الخارجية والتي بدورها تتكرر ثلاث مرات بمعنى أن الحلقة الداخلية تتكرر 12 مرة

وخرج البرنامج هو التالي :

```
1  2  3 4  2  4 6 8  3 6
```

## 2- جمل (while) المتداخلة

تشبه حلقات while حلقات for المتداخلة

وتأخذ الشكل العام التالي :

```
while (   condition  )
   {   Statements
       while  ( condition  )
       { ……….
         ……….
        while ( condition  )
        {     Statements
         }
         ……..
       }
      ……
   }
```

لنكتب البرنامج السابق باستخدام ( while )

```
# include < iostream .h>
main(    )
{   int  i  ,  j  =1;
   while (  i <= 3 )
    {
      while  ( j<=4)
       {
          cout  << i*j<<"     "<< endl;
            j++;
          } // end inner while
      i++;
      j=1;
```

```
    } // end outer while
    return 0 ;
} // end main
```

## محاضرة(8)
## تكملة الحلقات المتداخلة

**مثال :**

حساب وطباعة المضروب (factorial ) للأعداد من واحد إلى عشرة حيث أن
مضروب العدد هو (n!=n(n-1)(n-2) . . . 3*2*1).
مثلاً مضروب العدد (7) هو (7!=7*6*5*4*3*2*1 )

**البرنامج** :

```
#include<iostream.h>
   int main()
       {
          long factorial ;
          int  k,a =1;
          while(a<=10) //while 1
          {
                   factorial=1;
                    k=1;
                   while(k<=a) //while 2
                   {
                      factorial*=k;
                      ++k;
                      } //end while 2
                 cout <<a<<"!= "<< factorial<<endl;
                 ++a;
        } //end while 1
        return  0  ;
} // end main
```

شكل الإخراج على الشاشة جراء تنفيذ البرنامج :

```
1!=1
2!=2
3!=6
4!=24
5!=120
6!=720
```

**شرح كيفية عمل البرنامج :**

| a | factorial | k | |
|---|---|---|---|
| 1 | 1 | 1 | |
| | 1 | 2 | |

يظهر على الشاشة السطر التالي 1!=1

| | | | |
|---|---|---|---|
| 2 | 1 | 1 | |
| | 1 | 2 | |
| | 2 | 3 | |

يظهر على الشاشة السطر التالي 2!=2

| | | |
|---|---|---|
| 3 | 1 | 1 |
| | 1 | 2 |
| | 2 | 3 |
| | 6 | 4 |

يظهر على الشاشة السطر التالي 3!=6

| | | |
|---|---|---|
| 4 | 1 | 1 |
| | 1 | 2 |
| | 2 | 3 |
| | 6 | 4 |
| | 24 | 5 |

يظهر على الشاشة السطر التالي 4!=24

## 3- جمل (do... while ) المتداخلة

**مثال :**

برنامج حساب المضروب للأعداد من واحد إلى عشرة وطباعة هذا المضروب باستخدام حلقة التكرار المتداخلة (do..while )

```
#include<iostream.h>
int main()
{
  long factorial;
```

```
  int k ,a  =1;
  do
  {
      factorial=1;
      k=1;
      do
      {
          factorial*=k;
          ++k;
       } while (k<=a)
        cout<<a<<"!="<< factorial<<endl;
         ++a;
  }  while (a<=10);
   return 0;
}
```

## جمل أخري

### 1- جملة الايقاف ( break )

وظيفة break هي ايقاف بنية أو حلقة تكرار عند تحقق شرط أو شروط معينة وعند تنفيذها يتم القفز إلى سلسلة الجمل التالية للبنية أو حلقة التكرار .

**مثال**

```
# include < iostream .h >
Main (  )
{  int I ,j;
  for ( i=1; i<=100; i++)
    {  cout << i;
```

```
    If ( i== 10)    break;
   }
Return 0;
 }
```

يتوقف تنفيذ الحلقة for عندما يصبح ( i=10 ) ويتم الخروج منها

### 2- جملة الاستمرار ( continue )

تعمل (continue ) على تجاوز تنفيذ بقية الجمل في التكرار خلال الحلقة الحالية والانتقال الى بداية الحلقة من جديد

```
# include < iostream .h>
Main( )
{ int  x,n ;
Do
    { cin >>x>>n;
       If ( n <1 ) continue ;
          Cout << x;
           n--;
}  while ( n<1)
Return 0;
}
```

تعمل continue على تجاوز تنفيذ الجملتين التاليتين لها وتبدأ حلقة جديدة اذا تحقق الشرط

### 3- جملة الخروج ( exit )

تعمل هذه الدالة على ايقاف البرنامج في مكان منه والخروج منه وتعني الخروج نهائياً من البرنامج وترجع القيمة (0) اذا انتهى البرنامج بنجاح أو أي قيمة غير الصفر اذا كان هناك خطأ .

```
# include < iostream .h >
  Int main (  )
  {   int  i ,number ,sum=0;
      For ( i=1;i<10;i++)
       { cout <<" Enter value : ";
          Cin>> number;
       if( number <0)
          {  cout <<"this is negative !";
          Exit(  );
```

```
        } // end if
       Sum + = number ;
    }// end for
Cout <<  "the sum of bositive numbers is :"<<sum;
Return 0;
} // end main
```

## 4- جملة اذهب ( goto )

جملة اذهب الى ( goto ) تستخدم لتحويل المسار التتابعي لأوامر البرنامج الى جملة معينة وتمتلك الصيغة التالية : goto lable;

Lable اسم العنوان وينطبق عليه نفس شروط اسم المتغير حيث يمكن وضعه في أي مكان من البرنامج

```
# include < iostream .h >

  Int main (    )
   {  int n  =10;
      Last : cout<<n<<"   ";
       n--;
        if ( n>0) goto    last  ;
       cout<<  " the number < 0";
return 0;
}
```

# المحاضرة ( 9 )

# الدوال ( functions )

في السابق كتبنا البرامج على انها كتلة واحدة فالبرنامج عبارة عن تعليمات متتالية يتم تنفيذها تباعاً من بداية البرنامج حتى نهايته لاكن في بعض الأحيان يكون هذا غير مناسباً حيث يصعب متابعة هذه البرامج وتصحيح الأخطاء الناجمة عن كتابة البرنامج بهذا الشكل لذلك يفضل تجزئة البرانامج إلى أجزاء صغيرة كل جزء من هذه الأجزاء يؤدي مهمة محددة حيث ترتبط هذه الأجزاء لتكون برنامجاً واحداً وهذا ما نسميه باسلوب أو مفهوم البرمجة التركيبية structured programming

## 1. الدالة function :

الدالة هي جزء من أجزاء البرنامج وتتكون من جملة واحدة أو مجموعة جمل ( تعليمات ) تكون كتلة واحدة ولكل دالة مهمة معينة تقوم بتنفيذها ضمن البرنامج ويمكن استدعائها من أكثر من نقطة في البرانامج .

البرنامج: بمفهوم البرمجة التركيبية في C++ عباره عن مجموعة من الدوال , إحدى هذه الدوال هي الدالة الرئيسية main ( ) والتي يبدأ بنتفيذ البرنامج من عندها وكذلك يتم استدعاء الدوال منها لأداء مهمة معينة ومن ثم الرجوع اليها كذلك . كما يمكن ان تستدعي الدوال بعضها البعض.

## 2. الصيغة العامة للدالة ( الإعلان أو التعريف عن دالة )

```
Return_type function _name (type argument ,type argument2….)
{
    Local variables
    Function bady
  }
```

**مثال**

```
int sum ( int num1  , int num2)
{
   int thesum;
   Thesum = num1+num2;
return (thesum);
}
```

Return_type يمثل نوع قيمة الدالة عند رجوعها الى المكان المستدعي لها مثلاً :

int أو double أو void

Function_name اسم الدالة حيث يخضع إختيار هذا الإسم لشروط تسمية المتغيرات

argument1,argument2….. معاملات استقبال البيانات الداخلة الى الدالة

Local variables وهي المتغيرات المحلية الخاصة بالدالة فالمتغيرات المحلية هي المتغيرات التي يعلن عنها وتستخدم في حدود الدالة ولايمكن التعرف عليها خارج حدود هذه الدالة. في هذا الجزء يتم الإعلان عن متغيرات الدالة مثلاً

```
int num1;

double num2;
```

Function body يمثل جسم الدالة ويتكون من جملة واحدة أو أكثر من جملة وفي حالة أن Return_type غير void لابد من كتابة الجملة في نهاية جسم هذه الدالة

```
return (expression) ;
```

### 3. فوائد الدوال

من فوائد الدوال ما يلي:

- عند تجزئة البرنامج إلى أجزاء صغيرة ( دوال ) تسهل عملية متابعة البرنامج وتصحيح الأخطاء الموجودة فيه من قبل المبرمج.
- الدوال تعمل على تلافي تكرار الجمل في أكثر من مكان في البرنامج فيتم كتابة دالة بها الجمل التي من الممكن أن يتم تكرارها في البرنامج ومن ثم تستدعي هذه الدالة من اي مكان في البرنامج.
- توفر الدوال الجاهزة الموجودة في مكتبة C++ الجهد والوقت عن طريق استخدام هذه الدوال وعدم اعادة كتابتها مرة أخرى.

## ملاحظات

- عادة يتم الاعلان أوالتعريف عن أي دالة قبل الدالة الرئيسية main ( )
- يتم استدعاء الدالة داخل البرنامج عن طريق اسمها فقط و اعطاءها قيم للمعاملات إذا كانت الدالة تمتلك معاملات.
- ينبغي كتابة return-type عند تعريف الدالة إذا كانت ترجع قيمة من النوع غير الصحيح ( )int , وفي حالة عدم كتابة return – type ( نوع القيمة الراجعة من الدالة ) يكون هذا النوع int تلقائياً ويستحسن كتابة return-type في كل الأحوال.
- قد يكون للدالة مهمة معينة تؤديها بدون ارجاع قيمة عند إنتهائها في هذه الحالة يعلن عنها من النوع الحالي ( void ) مثلاً :

```
void printMessage ( )
```

```
{
  cout <<" the function is : printMessage "<< endl;
}
```

### 4. برنامج يستخدم دالة لإجراء عملية جمع عددين صحيحين :

```
// function examples
# include < iostream .h>
int addition (int a , int b )
  {  int  c ;
     c = a+b;
     return (c);
  }
int main (  )
 {   int d ;
     d= addition (5,6);
     cout << "the result is "<<d;
     return(0);
  }
```

قمنا بتعريف دالة الجمع (addition ) والتي لها معاملات صحيحة ( a,b ) ثم قمنا بكتابة جسم الدالة المحتوي على الجمل الموجودة داخل الحاصرتين { } ثم استخدمنا متغير محلي ( c ) من النوع int لاسناد مجموع العددين ( a,b ) قمنا باستدعاء الدالة ( addition) من داخل الدالة الرئيسية ( ) main عن طريق اسمها فقط واعطاءها قيم ابتدائية

```
addition ( 5,6);
```

واسناد القيمة الراجعة من الدالة ( addition ) الى المتغير المحلي ( d ) للدالة ( )main

```
d= addition ( 5,6);
```

### 5. برنامج لادخال عددين واخراج هذين العددين بالترتيب التصاعدي واخراج مجموع هذين العددين باستخدام الدوال :

```
#include < iostream .h>
  int swap ( int x,int y)
```

```
  { int temp ;
    Temp=x;
     x=y ;
     y = temp;
    }
 int sum( int x,int y)
  { int result ;
     result = x+y;
     return result;
   }
 void printxyz (int x,int y,int res)
 {
   cout <<" the numbers are :  "<<x<<"      "<< y<< endl;
   cout << " the sum of number is :  "<<res;
  }
int main(   )
 {  int a,b, summ ;
    cout <<"enter tow numbers"<<endl;
    cin>>a>>b;
   If (a>b)
   Swap(a,b);
   summ = sum( a,b);
   printxyz(a,b,summ);
 }
```

## 6. الدوال الرياضية math functions

في لغة c++ توجد مجموعة من الدوال الجاهزة والموجودة ضمن المكتبة القياسية للغة ومن هذه الدوال: الدوال الرياضية الموجوده ضمن الملف ( math.h ) وفيما يلي بعض هذه الدوال :

| الدالة | الغرض منها | نوع المتغير | نوع القيمة المرجعة |
|---|---|---|---|
| abs (x) | القيمة المطلقة للعدد x | int | int |

| | | | |
|---|---|---|---|
| fabs(y) | القيمة المطلقة للعدد y | double | double |
| sqrt(x) | الجذر التربيعي للعدد x | double | double |
| pow(x,y) | X بقوة y | double | double |
| sin(x) | دالة الجيب | double | double |

## 7. المتغيرات الخارجية ( global variables )

تعرفنا سابقاً على المتغيرات المحلية والتي يعلن عنها وتستخدم داخل حدود الدالة ولا علاقة لها بالدوال الأخرى أما المتغيرات الخارجية فهي التي تكون معروفة لجميع الدوال الموجودة في البرنامج حيث يتم الاعلان عنها خارج كل الدوال وقبل الدالة الرئيسية ()main ولا يجوز الاعلان عنها أكثر من مرة.

**مثال** : اكتب برنامج يعمل الآتي :

a. ادخال نتيجة الامتحان الأول والثاني لكل طالب في فصل دراسي يضم أربعة طلاب
b. استخدام دالة لحساب أكبر درجة ومعدل وحالة كل طالب على النحو التالي :
   - ناجح (pass ) اذا كان معدل الطالب أكبر من أو يساوي 50
   - راسب ( fail ) اذا كان معدل الطالب أقل من 50

```
# include < iostream .h>
    float max , avg;
    String status;
    Calculate( float m1, float m2)
    { if (m1>m2)
        max =m1 ;
     else
        max= m2;
    avg=(m1+m2)/2;
    If (avg>=50)
        status="pass";
    else
```

```
        status="fail";
     }
   Print_them( )
  {  cout<<" the result : maximum = " <<max<<endl;
     cout <<"average= "<<avg<<endl;
     cout<<"status= "<<endl;
  }
int main (  )
  { int i ;
    float    t1 ,t2;
    for ( i=1;i<=4 ;i++)
    {   cout<<"number of student :"<<i<< endl;
        cout <<"Enter two grades:";
        Calculate(t1,t2);
        Print_them(t1,t2);
     }
     return0;
    }
```

## محاضرة (10)

## الاستدعاء الذاتي للدوال (Recursion)

المقصود بالإستدعاء الذاتي للدوال هو أن تقوم الداله باستدعاء نفسها بنفسها, فهناك الكثير من المسائل التي يمكن حلها باستخدام الاستدعاء الذاتي لداله ما. في هذه الحالة يمكن توفير الكثير من جمل التكرار. في لغة C++ يمكن أن تستدعي الدالة نفسها أو أن تستدعي دالة أخرى في نفس البرنامج.

**مثال**: اكتب برنامجاً يستخدم الاستدعاء الذاتي لحساب مضروب العدد ( factorial )

```
// factorial  calculater

 # include <iostream.h>
```

```
long factorial (long a)
 {
  if ( a>1 )
  return( a*factorial (a-1));
  else
  return(1);
 }
int main( )
 { long number;
  cout<<"please inter a number: "<<endl;
  cin>>number;
  cout<<number <<" != "<<factorial ( number);
  return 0;
 }
```

في هذا البرنامج انشأنا داله بإسم () factorial تقوم بحساب مضروب عدد عن طريق استدعاء نفسها عده مرات عند تحقق شرط معين .

فمثلاً عندما يكون العدد a=5 :

الاستدعاء الأول : factorial (5)=5*factorial(5-1)=5*factorial(4)

الاستدعاء الثاني: =5*4*factorial(3)

الاستدعاء الثالث : =5*4*3*factorial(2)

الاستدعاء الرابع: =5*4*3*2*factorial(1)

=5*4*3*2*1

والناتج النهائي سيكون 5*4*3*2*1 =120

**يمكن تغيير الداله ()factorial في البرنامج السابق ليصبح الرنامج بالشكل التالي:**

```
 # include <iostream.h>
  long factorial (long a)
   {
    if (a==0 || a== 1)
        retutn (1);
    else
       return( a*factorial (a-1));
   }
int main( )
 {  long number;
  cout<<"please inter a number: "<<endl;
  cin>>number;
  cout<<number <<" != "<<factorial ( number);
  return 0;
 }
```

## المحاضرة (11)

## المصفوفات (Arrays )

نعرف أن اسم المتغير يحمل قيمة واحدة فقط قابلة للتغيير اثناء تنفيذ البرنامج ,فطرق التعامل مع أسماء المتغيرات و التوابت المختلفه التي وردت سابقا تعد صالحة للتعامل مع عدد محدود من هذه المتغيرات و الثوابت , سواء في عمليات الإدخال و الإخراج أو العمليات الحسابيه و المنطقيه. فعندما يصبح لدينا مجموعة كبيره من البيانات التي لها نفس النوع أي أن عدد القيم الثابته أو المتغره التي نريد التعامل معها كبيرا نسبيا تصبح تلك الطرق غير عمليه. هذه المجموعة من القيم تحتاج إلى عملية تخزين في متغير واحد لكي يسهل التعامل مع هذه القيم لذالك يتم استخذام مفهوم المصفوفه.

**تعريف المصفوفة** : هي عبارة عن سلسلة من العناصر التي لها نفس النوع والموضوعة في الذاكرة بشكل متجاور والتي تحمل اسماً واحداً بحيث يسهل الرجوع إلى أي من هذه العناصر عن طريق رقم هذا العنصر index في هذه السلسله فالعنصر الأول في المصفوفه يمتلك الرقم صفر و العنصر الثاني يمتلك الرقم 1 و هكذا .., نستطيع التعامل مع عناصر المصفوفه عن طريق اسم المصفوفه و أرقام عناصرها في داخل هذه المصفوفة.

**تنقسم المصفوفات إلى قسمين هما:**

1- مصفوفات ذات البعد الواحد one dimensional arrays
2- مصفوفات متعددة الابعاد multi-dimensional arrays

في لغة الـ C++ يوجد الأمر #define الذي يكتب عادة بعد أوامر #include وهذه الأوامر يتم تنفيذها قبل البدء بترجمة البرانامج وتسمى أوامر ماقبل المعالجة preprocessor directive , الامر #define يقوم بانشاء معرفات تسمى الثوابت الرمزية (symbolic constant) في كثير من البرامج يستخدم هذا الأمر في الإعلان عن عدد عناصر المصفوفة .

الصيغة العامة للامر # define constantName constantValue

**<u>مثال</u>**

```
# define maximum 13

# define name  "Nashwan"
```

## 1. المصفوفات ذات البعد الواحد One Dimensional Arrays

**الصيغه العامه للإعلان عن مصفوفه ذات بعد واحد:**

```
Type arrayName [size];
```

حيث :

Type - يمثل نوع البيانات في المصفوفة مثل int, double, float و غيرها

ArrayName – يمثل إسم المصفوفة الذي نقوم بإختياره و هذا الاسم يخضع لشروط تسمية المعرفات.

Size – عدد عناصر المصفوفة

**مثلاً** : int numbers [ 6 ];

هنا نعلن عن مصفوفة اسمها numbers عدد عناصرها = 6 هذه العناصر من النوع الصحيح int

الشكل التالي يظهر كيف يتم حجز مواقع في الذاكرة لعناصر المصفوفة numbers

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|

هذه الارقام تبين ارقام عناصر هذه المصفوفه و ليس قيم هذه العناصر

حيث تحجز مساحة في الذاكرة للمصفوفة حسب نوع البيانات المخزنة (تكلمنا سابقا عن حجم البيانات ي الذاكره بالبايت للأنواع المختلفه من البيانات) في المصفوفة فتحجز خانات متجاوره كل خانة من هذه الخانات ترقم برقم, الترقيم يبدأ من الصفر ويسمى رقم العنصر بالدليل (index ) فالعنصر الاول من المصفوفة دليله (0) والعنصر الثاني من المصفوفة دليله (1) والعنصر الثالث من المصفوفة دليله (2) وهكذا... .

| | numbers[0] | numbers[1] | numbers[2] | numbers[3] | numbers[4] | numbers[5] |
|---|---|---|---|---|---|---|
| numbers | | | | | | |

**القيم الابتدائية للمصفوفة ذات البعد الواحد (initial values )**

يمكن تخصيص قيم ابتدائية لمصفوفة البعد الواحد بطريقتين :

الاوليً : اثناء الاعلان عن المصفوفة والصيغة العامة هي :

Type arrayName [size]={ value1,value2 ,……,valueN};

( value1,value2,…….,valueN) : تمثل قيم المصفوفة على الترتيب , هنا عدد العناصر هو N والذي يساوي قيمة ( size )

**مثال** : لتكن لدينا المصفوفة :

int numbers[6]={2,5,10,14,18,0};

إذاً

numbers[o]=2 , numbers[1]=5, numbers[2]=10, numbers[3]=14,numbers[4]=18, number[5]=0

الثانية : اثناء تنفيذ البرنامج بكتابة اسم عنصر المصفوفه وإعطاؤه قيمه مثلا :

numbers[0]=37;

لنكتب برنامجاً يقوم بتنفيذ الخطوات السابقه (يعرف مصفوفه مكونه من 6 عناصر من النوع الصحيح int و يعطيها القيم: {2,5,10,14,18,0} ) ثم يقوم بطباعه عناصر المصفوفه و مجموع هذه العناصر.

```
#include <iostream.h>

# define size  6

int main ()

{

        int numbers [size] = {2,5,10,14,18,0};

        int n , sum=0;

       for (n=0; n < size ; n++)

                    {


                       cout << "numbers["<<n<<"]:  "<<numbers[n]<<endl;

                       sum += numbers[n];

                     }

        cout << "The sum of elements is:  "<< sum;

}
```

**مخرجات البرنامج :**

numbers[0]:  2

numbers[1]:  5

numbers[2]:  10

numbers[3]: 14

numbers[4]: 18

numbers[5]: 0

The sum of elements is: 49

# المحاضرة (12)

# تكملة للمصفوفات (Arrays )

## 2. المصفوفات ذات البعدين Two Dimensional Arrays

المصفوفات ذات البعدين ( two dimenstional arrays ) تتكون من صفوف وأعمدة

**الصيغة العامة للإعلان عن مصفوفة ذات بعدين**

Type arrayName[size1][size2]

حيث أن:

Type : نوع البيانات في المصفوفة

arrayName :اسم المصفوفة

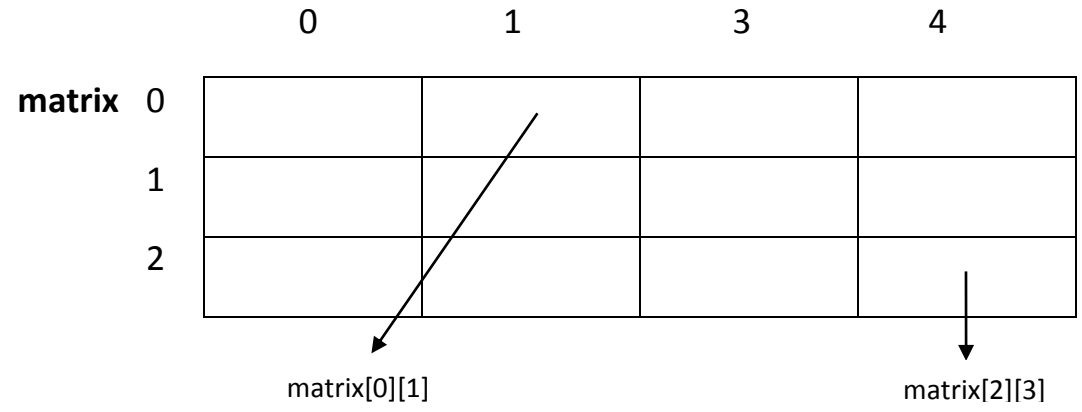
size1 : عدد الصفوف في المصفوفة

size2 : عدد الأعمدة في المصفوفة

**مثال** : int matrix [ 3][ 4] ;

في هذا المثال تم الاعلان عن مصفوفة اسمها matrix والتي تتضمن 3 صفوف ( اسطر) واربعة اعمدة أي أن عدد عناصر المصفوفة = 12 عنصراً يبدأ ترقيم عناصر المصفوفة من الرقم 0 ويسمى هذا الرقم الدليل ( index ) فالعنصر الاول في المصفوفة الثنائية يكون دليلة [0][0] والعنصر الواقع في السطر الأول وفي العمود الثاني يكون دليله [1][0] وهكذا

الشكل التالي يوضح دليل عناصر والمصفوفة الثنائية :




**القيم الابتدائية للمصفوفة ذات البعدين (initial values )**

يمكن تخصيص قيم ابتدائية لمصفوفة ذات بعدين بطريقتين

الاوليً : اثناء الاعلان عن المصفوفة والصيغة العامة هي :

Type arrayName[size1][size2]={{list of first row elements},{list of second row elements},...,{list of last row elements}};

**مثال** : لتكن لدينا المصفوفة :

Int matrix[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};

إذاً

matrix[o][0]=1 , matrix[0][1]=2, matrix[0][2]=3, matrix[0][3]=4

matrix[1][0]=5 , matrix[1][1]=6, matrix[1][2]=7, matrix[1][3]=8

matrix[2][0]=9 , matrix[2][1]=10, matrix[2][2]=11, matrix[2][3]=12

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 |
| 1 | 5 | 6 | 7 | 8 |
| 2 | 9 | 10 | 11 | 12 |

الثانية : اثناء تنفيذ البرنامج بكتابة اسم عنصر المصفوفه و إعطاؤه قيمه أو بإستخدام دالة الإدخال cin

**مثال1** :

matrix [2][1]=63;

**مثال2** :لنكتب برنامجا يقوم ب لنكتب برنامجاً يعرف مصفوفه مكونه من 3 أسطر و 4 أعمدة عناصرها من النوع الصحيح int ثم يقوم بإدخال قيم عناصر هذه المصفوفه بإستخدام الداله cin ثم يقوم بطباعة قيم عناصر هذه المصفوفه و مجموع هذه القيم.

```
# include <iostream.h>
#define rows 3
#define col 4
int main (  )
  {  int matrix [rows][col];
     int  i, j, sum=0
     cout<<"input the elements of array:"<<endl;
     //Input the elements of array
     for(i=0; i<rows; i++)
```

```
        for(j=0; j<col; j++)
            cin>> matrix[i][j];
      //Output the elements of array
      for (i=0; i<rows; i++)
        for(j=0;j<col;j++)
          {
           cout<<"matrix["<< i <<"]["<< j <<"]= " << matrix[i][j] <<endl;
            sum += matrix[i][j];
           }
     cout << "The sum of elements is:  "<< sum;
     return0;
}
```

## 3. المصفوفات الحرفية ( character arrays )

في لغة الـ C++ يمكن التعامل مع السلاسل الحرفية وكأنها مصفوفات يسمى هذا النوع من المصفوفات بالمصفوفات الحرفية.
مثلاً الكلمة welcome يمكن التعامل معها بشكل مصفوفة حرفية كما يلي:

```
char greeting [7]={'w','e','l','c','o','m','e','\0'};
```

حيث يتم الاعلان عن مصفوفة من النوع char واسمها greeting وتتصمن القيم التالية: 'w','e','l','c','o','m','e' بالاضافة الى الرمز '\0' والذي يسمى بالرمز الصفري , يجب ان يكون هذا الرمز موجود في نهاية المصفوفة الحرفية والغرض من هذا الرمز الصفري هو معرفة مترجم اللغة بنهاية عناصر المصفوفة وعليه يجب حجز مكان له في الذاكرة , أي ان دليل المصفوفة (index ) يضاف له 1 عند الاعلان عن المصفوفة الحرفية, فعدد عناصر المصفوفة السابقة عند الاعلان يساوي 7 وليس 6. بمعني أن السلسلة الحرفية يمكن كتابتها علي شكل مصفوفة ولكن يجب الانتباه الى حجم

المصفوفة الذي يجب ان يكون بعدد احرف السلسلة الحرفية مضاف اليه 1 والذي يكون مخصص للرمز الصفري '\0' فيمكن كتابة المصفوفة الحرفية بالشكل التالي:

```
char greeting [7] = "welcome";
```

ويمكن عدم ذكر حجم المصفوفة اثناء الاعلان عنها كما في الشكل التالي:

```
char greeting [ ]="welcome";
```

**مثال** :

```
# include <iostream.h>
int main( )
{
  char  yourName [ ]= "Please, Enter your first name :  ";
  char  welcome [ ]= "Hello";
  char  name[20];
  cout <<  yourName ;
  cin >> name;
  cout <<  welcome << “,  Mr: ” << name<<"\n";
  return 0;
}
```

## المحاضرة (13)

## العمليات على المصفوفات 0prations on arrays

يمكن تنفيذ الكثير من العلميات على المصفوفات مثل العمليات الحسابية المختلفة المختلفة.

**مثال1** : إكتب برنامجاً يقوم بادخال مصفوفتين من النوع الصحيح ثم يقوم باجراء عملية الجمع على هاتين المصفوفتين و يخرج المصفوفه الناتجه عن عمليه جمع المصفوفتين.

$$\dot{a} = \begin{matrix} 2 & 3 & 4 \\ 5 & 6 & 7 \\ 8 & 9 & 10 \end{matrix} \quad , \quad \dot{b} = \begin{matrix} 11 & 12 & 13 \\ 14 & 15 & 16 \\ 17 & 18 & 19 \end{matrix}$$

**البرنامج:**

```
# include <iostream.h>
# define Row  3
# define Col 3
int main ()
{
int arr1 [Row][Col],  arr2 [Row][Col],  sumArr [Row][Col];
int i, j, k;
for (i=0; i < Row; i++)
     for (j=0; j < Col; j++)
      {
          هنا يطلب البرنامج من المستخدم ادخال قيمه عنصرين للمصفوفتين مثلا: قيمة العنصر
          الأول في المصفوفه الأولي و قيمة العنصر الأول ايضاً في المصفوفه الثانيه :
        cout << “Enter two elements for two arrays: ” ;
        cin >> arr1[i][j] >> arr2[i][j]
      }
for (i=0; i < Row; i++)
     for (j=0; j < Col; j++)
         sumArr [i][j] = arr1[i][j] + arr2[i][j];
cout << “The sum of two arrays: “ << endl;
for (i=0; i< Row; i++)
   {
     for  (j=0; j< Col; j++)
          cout << sumArr[i][j] << “\t”;
     cout << endl;
     }
```

```
return 0;
}
```

## مخرجات البرنامج

في البدايه سيطلب منا البرنامج إدخال عناصر المصفوفتين و سيدخل المستخدم قيما لتخزينها في المصفوفه و لنفرض أن المستخدم سيدخل القيم التالية:

```
Enter two elements for two arrays: 2 11
Enter two elements for two arrays: 3 12
Enter two elements for two arrays: 4 13
Enter two elements for two arrays: 5 14
Enter two elements for two arrays: 6 15
Enter two elements for two arrays: 7 16
Enter two elements for two arrays: 8 17
Enter two elements for two arrays: 9 18
Enter two elements for two arrays: 10 19

The sum of two arrays:
13      15      17
19      21      23
25      27      29
```

**مثال 2**: ليكن لدينا المصفوفه التالية:

$$a = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

إكتب برنامجاً يقوم بادخال عناصر هذه المصفوفة ثم إيجاد و طباعة التالي:

1. حاصل ضرب عناصر القطر الرئيسي لهذه المصفوفة, أي إيجاد: (a[0][0]*a[1][1]*a[2][2])
2. أكبر عنصر في المصفوفة.

**البرنامج:**

```
# include <iostream.h>
# define Row 3
# define Col 3
int main ( )
{
  int a[Row][Col]= {{1,2,3},{4,5,6},{7,8,9}};
  int i, j, max, product;
  max = a[0][0];
  product = 1;
  for (i=0; i< Row; i++)
      for (j=0; j< Col; j++)
      {
        if (i==j)
           product *= a[i][j];
        if (a[i][j] > max)
           max = a[i][j];
       }
```

```
    cout << “The product is: ” << product << endl;

    cout << “Maximum element in array is: ”<< max;

    return 0;

}
```

**مخرجات البرنامج**

```
The product is: 45

Maximum element in array is: 9
```