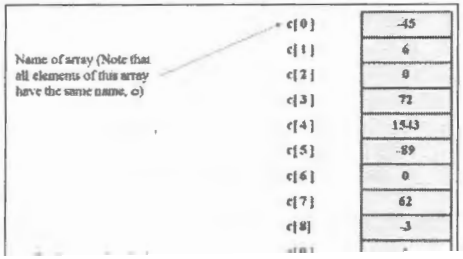
معنـي المصفوفة

كما تعلم من الرياضيات أن المصفوفة Array عبارة عن مجموعة من القيم (العناصر) من نفس النوع و تأخذ أسم و تأخذ عدد للعناصر والمطلوب كيفية تمثيلها وإستعمالها وكيفية التعامل مع عناصرها وللمصفوفة تطبيقات كثيرة لا تصلح إلا بها مثل عمليات البحث عن قيمة ضمن مجموعة قيم Search ومثل عمليات ترتيب مجموعة من القيم سواء تصاعدى أو تنازلى Sorting ومثل ايجاد اكبر قيمة ضمن مجموعة قيم Max وايجاد أقل قيمة Min والكثير من التطبيقات وخاصة فى مجال الدرسات العليا والابحاث

أنواع المصفوفات :

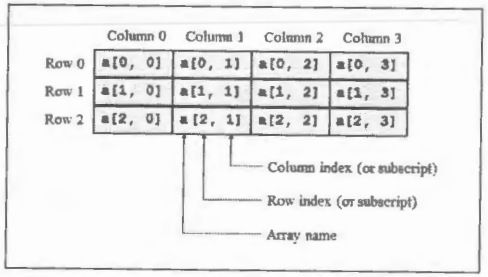
مصفوفة بعد واحد One Dimension كما بالشكل

A= [5 4 7 8 4 3]



مصفوفة متعددة الأبعاد :

و فيها يوجد أكثر من صف Row وأكثر من عمود Column كما بالشكل .



وفي هذه المصفوفة يوجد 6 صفوف Rows و3 أعمدة Columns لذلك تسمي 6*3

ويتم ترتيب العناصر بالصف Row ثم العمود Column كما بالشكل التالي :

C (0, 0) العنصر الأول الذي يقع في الصف الاول رقم 0 و العمود الاول رقم 0

C (0, 1) العنصر الثاني الذي يقع في الصف الاول رقم 0 و العمود الثانى رقم 1

C (0, 2) العنصر الثالث الذي يقع في الصف 0 و العمود 2

C (0,3) العنصر الرابع الذي يقع في الصف 0 و العمود 3

C (1,0) العنصر الخامس الذي يقع في الصف 1 و العمود 0

C (1,1) العنصر السادس الذي يقع في الصف 1 و العمود 1

و هكذا حتى باقي العناصر

ولإنشاء مصفوفة نستعمل ثلاثة خطوات هى:

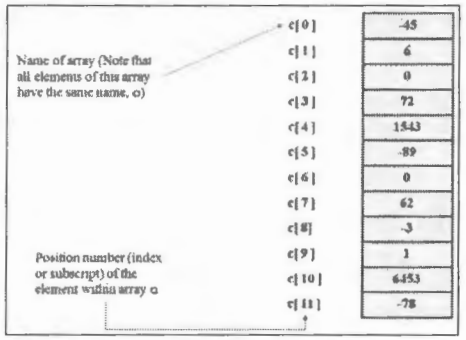
معني المصفوفة

كما تعلم من الرياضيات أن المصفوفة Array عبارة عن مجموعة من القيم (العناصر) من نفس النوع و تأخذ أسم و تأخذ عدد للعناصر والمطلوب كيفية تمثيلها وإستعمالها وكيفية التعامل مع عناصرها وللمصفوفة تطبيقات كثيرة لا تصلح إلا بها مثل عمليات البحث عن قيمة ضمن مجموعة قيم Search ومثل عمليات ترتيب مجموعة من القيم سواء تصاعدى أو تنازلى Sorting ومثل ايجاد اكبر قيمة ضمن مجموعة قيم Max وايجاد أقل قيمة Min والكثير من التطبيقات وخاصة فى مجال الدرسات العليا والابحاث

أنواع المصفوفات :

مصفوفة بعد واحد One Dimension كما بالشكل

```
A= [5 4 7 8 4 3]
```

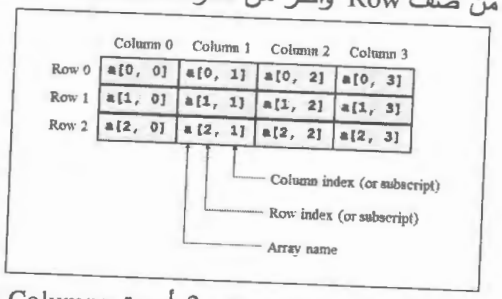


المصفوفة A مصفوفة صف One Row وهي ذات بعد واحد حيث لا يوجد صفوف وأعمدة بل هي صف واحد فقط .

المصفوفة B مصفوفة عمود واحد One Column ، لذلك يسمي كلا منهما مصفوفة البعد الواحد ويتم ترتيب العناصر بأسم المصفوفة وترتيب العنصر مثل A[1] , B[2] وهكذا .

مصفوفة متعددة الأبعاد :

و فيها يوجد أكثر من صف Row وأكثر من عمود Column كما بالشكل .



وفي هذه المصفوفة يوجد 6 صفوف Rows و3 أعمدة Columns لذلك تسمي 3*6

ويتم ترتيب العناصر بالصف Row ثم العمود Column كما بالشكل التالي :

C (0, 0) العنصر الأول الذي يقع في الصف الاول رقم 0 و العمود الاول رقم 0

C (0, 1) العنصر الثاني الذي يقع في الصف الاول رقم 0 و العمود الثانى رقم 1

C (0, 2) العنصر الثالث الذي يقع في الصف 0 و العمود 2

C (0,3) العنصر الرابع الذي يقع في الصف 0 و العمود 3

C (1,0) العنصر الخامس الذي يقع في الصف 1 و العمود 0

C (1,1) العنصر السادس الذي يقع في الصف 1 و العمود 1

و هكذا حتى باقي العناصر

ولإنشاء مصفوفة نستعمل ثلاثة خطوات هى:

- الإعلان عن متغير المصفوفة Array Variables
- تعريف (إنشاء) عنصر المصفوفة array object
- تخزين القيم أو البيانات داخل المصفوفة والتعامل معها

1- الإعلان عن متغير مصفوفة Decrying Array Variable

ويتم ذلك بطريقتين هما:

الطريقة الأولى:

```
String ourteam [];
int Codes [];
```

فى هذه الطريقة يتم كتابة نوع عناصر المصفوفة فى الأول (String , int) ثم كتابة اسم المصفوفة [our team, codes] ثم وضع الأقواس [ ] وبالتالى يتم الإعلان عن مصفوفة بالاسم ourteam من النوع string والذى يفرق بين المصفوفة والمتغير العادى هو الأقواس [ ] التى تدل على أن اسم المتغير هو مصفوفة.

الطريقة الثانية:

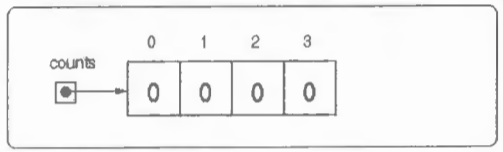
```
String [ ] ourteam;
int [ ] codes;
```

فى هذه الطريقة يتم كتابة نوع المصفوفة ثم الأقواس بالشكل [ ] string ثم اسم المصفوفة وهى our team فى السطر الأول.

بالمثل المثال الثانى هو [ ] int ثم اسم المصفوفة وهو Codes

وفى الحالتين تم الاعلان عن متغير من نوع هذه المصفوفة

والشكل التالى يوضح كيف تمثل المصفوفة فى الذاكرة



تعريف عنصر المصفوفة Orating Array Object

بعد الإعلان عن متغير مصفوفة يتم تعريف (إنشاء) عنصر المصفوفة ويتم بطريقتين

- استعمال كلمة new لإنشاء العنصر
- تعريف عنصر object المصفوفة مع إعطائها قيم ابتدائية بدون استعمال new

استعمال new

ويتم استعمال new لتعريف object عنصر كما يلى:

```
tring jobs= new string [10]
```


وفى هذا السطر يتم تعريف متغير بالاسم jobs من نوع مصفوفات حرفيات string [ ] كما سبق حجز او تعريف مصفوفة عناصر جديدة باستعمال new فى الطرف الثانى من النوع string وعدد العناصر 10 وبالطبع يمكن إجراء هذا السطر على سطرين كما يلى:

```
String [ ] jobs;
jobs = new string [10];
```

وعند استعمال new مع المصفوفة لابد من تحديد عدد العناصر مثل 10.

تعريف عنصر المصفوفة وإعطائها قيم ابتدائية

يمكن تعريف عنصر مصفوفة بدون استعمال new وذلك بإعطاء المصفوفة قيم ابتدائية وهنا لا نحتاج لتحديد عدد العناصر لأننا نضع العناصر نفسها كما يلى:

```
String [ ] jobs = {"Eng", Doc', "Tech"};
```

فى هذا السطر تم تعريف مصفوفة حرفيات بالاسم jobs ثم إعطائها قيم ابتدائية وهى الكلمات ...Eng وفى هذه الحالة لا تحتاج تحديد عدد العناصر لان العناصر موجودة بالفعل.

فى حالة استعمال New وعند إعطاء المصفوفة قيم ابتدائية يتم تسجيل قيم افتراضية حسب نوع المصفوفة فمثلاً مصفوفة القيم الرقمية تأخذ العناص0 ومصفوفة الحرفيات تأخذ 0/ ومصفوفة القيم المنطقية Boolean تأخذ القيمFalse ومصفوفة العناصر Objects تأخذ null.

التعامل مع عناصر المصفوفة Accessing Array Elements

ويتم التعامل مع عناصر المصفوفة بكتابة رقم عنصر المصفوفة بين أقواس المصفوفة [ ]، كما بالشكل.

```
Array Name [ No]
```

وفى هذه الصورة Array Name اسم المصفوفة و no ترتيب العنصر فى المصفوفة مع ملاحظة أن ترتيب عناصر المصفوفة تبدأ من القيمة 0 ، وتقوم لغة Javaباختبار قيمة ترتيب العنصر قبل التعامل معه للتأكد من وجود هذا العنصر ضمن عناصر المصفوفة وليس خارجها ولا يمكن تسجيل قيمة فى عنصر غير موجود ضمن عناصر المصفوفة (كما يحدث فى لغة C/C++) فلا يحدث خطأ ولتوضيح ذلك تابع المثال التالى:

```
String [ ] My Array = new String [20];
        My Array [20] = "Java lang";
```

فى السطر الأول يتم الإعلان وتعريف مصفوفة بالاسم My Array من نوع حرفيات string وحجز 20 عنصر باستعمال new.

فى السطر الثانى يتم تسجيل العبارة .Java lang فى المصفوفة My Array فى العنصر رقم 20 وهذا السطر عند ترجمة البرنامج يعصى خطأ وذلك لان هذه المصفوفة تبدأ من العنصر رقم 0 وتنتهى عند العنصر 19 وعدد العناصر 20عنصر لذلك لا يوجد عنصر رقمه 20.

وللتعامل مع عدد عناصر المصفوفة يمكن استعمال الدالة LENGTH كما فى السطر التالى:

```
L= MyArray. Length;
```

فى هذا السطر يتم حساب عدد عناصر المصفوفة باستعمال الدالة Length مع المصفوفة وحفظ هذه العدد فى المتغير L.

مثال

فى هذا المثال نحاول تجميع النقاط السابقة عن المصفوفة وتوضيحها كما فى السطور التالية:


```
1  public class Myfriends {
2
3      String [ ] Names = { "khaled", "Alaa", "Amr", "Omr"};
4  void printNames ( )
5  {
6      int I = 0;
7      System.out.println (Names [I]);
8      I++;
9      System.out.println (Names [I]);
10     I++;
11 System.out.println (Names [I]);
12     I++;
13     System.out.println (Names [I]);
14 }
15 public static void main (String args [ ])
16 {
17
18 Myfriends FD= new Myfriends ( );
19 FD. printNames ();
20 }
21 }
```

فى هذه السطور:

فى السطر رقم 1 يتم تعريف فصيلة البرنامج (Class) بالاسم Myfriends .

فى السطر رقم 3 يتم تعريف مصفوفة حرفيات (string) بالاسم Names مع إعطائها قيم ابتدائية عبارة عن مجموعة أسماء.

فى السطر رقم 4 يتم تعريف دالة عضوه دالة فى الفصيلة بالاسم printNames()

فى السطر رقم 5 يتم الإعلان عن متغير I بقية ابتدائية 0

فى السطر رقم 6 يتم طباعة عنصر المصفوفة رقم i اى رقم 0 اى الاسم Khaled المخزن فى العنصر

فى السطر رقم 7 يتم زيادة قيمة المتغير i بمقدار 1 لتصبح I تساوى 1

فى السطر رقم 8 يتم طباعة عنصر المصفوفة رقم i اى رقم 1 اى Alaa وهكذا السطور حتى السطر رقم 12

وفى السطر رقم 13 تنتهى الدالة Print Names() التى تقوم بطباعة الأسماء

فى السطر رقم 15 تبدأ الدالة الرئيسية main() التى يبدأ التنفيذ منها وإذا كان هناك اكثر من فصلة class يتم تنفيذ الفصيلة التى بها الدالة main كما أشرنا من قبل.

فى السطر رقم 18 يتم تعريف عنصر (object) بالاسم FD فى الفصيلة My friends وذلك لنتمكن ممن التعامل مع أعضاء الفصيلة My friends فى دوال وبيانات

فى السطر 19 تم استدعاء الدالة ()PrintNames مع هدف الفصيلة FD وبالتالى تقوم الدالة بطباعة الأسماء المعرفة داخل المصفوفة Names وتحصل على قائمة بالأسماء كما فى نتيجة التنفيذ.

اكتب هذا البرنامج ونفذه تحصل على قائمة بالأسماء التى تم حفظها فى المصفوفة كما بالشكل

وفى المثال السابق تم طباعة عناصر المصفوفة داخل الدالة ()printNames عنصر بعنصر بينما لا يناسب ذلك الحالات العملية. فلو فرضنا ان عدد عناصر المصفوفة 100 عنصر ستضطر لكتابة 100 سطر لطباعة العناصر ، ولذلك يكون الحل هو استعمال جمل التكرار التى توفر ذلك كما سيلى فى هذا الفصل.

استعمال أوامر التكرار for مع المصفوفة

لاحظنا فى المثال السابق كيف تم التعامل مع عناصر المصفوفة بحديد ترتيب عنصر المصوفة كما فى السطر

```
Names[3]
```

اى العنصر رقم 4 ولكن أشرنا الى انه يصعب أو يستحيل استعمال هذه الطريقة مع المصفوفات ولابد من استعمال جملة لتكرار for بدلا من ذلك وبالتالى يتم تعديل البرنامج السابق ليصبح كما فى السطور التالية:

```
1  public class Myfriends {
2
3      String [ ] Names = { "Khaled", "Alaa", "Amr", "Omr"};
4  void printNames ( )
5  {
6    int i ;
7    for (i=0;i<4;i++)
8    System.out.println (Names [i]);
9      }
10   public static void main (String args [ ])
11   {
12
13   Myfriends FD= new Myfriends ( );
14   FD  printNames ();
15   }
16   }
```


الجديد فى هذه السطور عن برنامج المصفوفات التقليدى هو استعمال التكرار for Loop

فى السطر رقم 4 للتكرار من 0 حتى Names. Length وهو عدد عناصر المصفوفة

فى السطر رقم 5 يتم طباعة عناصر المصفوفة Names[I] اى العناصر المخزنة

وبالتالى يتم طباعة الأسماء وذلك يوفر إعادة كتابة أمر الطباعة وذلك باستعمال تكراره مع for بعدد عناصر المصفوفة وعند تنفيذ البرنامج تحصل على نفس النتيجة السابقة كما يتضح لنا أهمية استعمال جملة التكرار for مع المصفوفات .

نسخ المصفوفات

يمكن استعمال متغير مصفوفة للاشارة الى مصفوفة اخرى كما فى الشكل

```
double[] a = new double[3];
double[] b = a;
```

فى هذه السطور يتم الاعلان عن مصفوفة a ثم الاعلان عن مصفوفة b واستخدام متغيرها للاشارة الى نفس المصفوفة a

طول المصفوفة

```
for (int i = 0; i < a.length; i++)
    {
     b[i] = a[i];
    }
```

فى هذه السطور

يتم الاستفادة من الخاصية length الموجودة فى المصفوفة لتحديد طولها أى عدد عناصرها بدلا من تحديد هذا الطول برقم ثم القيام بنسخ عناصر المصفوفة a الى المصفوفة b

ايجاد أكبر قيمة

يمكن استعمال المصفوفات لايجاد أكبرقيمة من مجموعة قيم وذلك كما فى السطور التالية :

```
double max = A[0];
```

```
    for (int i = 1; i < A.length; i++) {
      if (A[i] > max)
        max = A[i];
    }
```

فى هذه السطور

يتم وضع قيمة اول عنصر من المصفوفة فى المتغير mx بافتراض انه أكبر عنصر ثم استعمال تركيب التكرار for فى المرور على جميع عناصر المصفوفة وفى كل مرة يتم مقارنة قيمة المتغير max بعنصر المصفوفة الحالى

اذا كانت القيمة الموجودة فى المصفوفة أكبر من قيمة المتغير mx يتم التبديل بحيث ينتهى التكرار وقد تم وضع أكبر عنصر فى المتغير mx

البحث عن قيمة داخل المصفوفة

من العمليات المشهورة للمصفوفات القيام بالبحث عن قيمة داخل مجموعة قيم والمثال التالى يوضح ذلك

```
static int find(int[] A, int N)
    {
      for (int index = 0; index < A.length; index++)
        {
          if ( A[index] == N )
        return index;  // N has been found at this index!
        }
          return -1;
    }
```

فى هذه السطور

يتم انشاء دالة تستقبل اسم المصفوفة والقيمة المطلوب البحث عنها ثم يتم استعمال تركيب التكرار for للمرور على العناصر ومقارنتها بالقيمة حتى نجدها

فاذا وجدت يتم اعادة ترتيب هذه القيمة فى المصفوفة والا يتم اعادة القيمة -1

ترتيب عناصر المصفوفة

من العمليات المشهورة أيضا استعمال المصفوفات فى ترتيب مجموعة قيم ولتوضيح ذلك تابع السطور التالية :


```
static void selectionSort(int[] A)
{
   for (int lastPlace = A.length-1; lastPlace > 0; lastPlace--)
       {
  position lastPlace.

          int maxLoc = 0;  // Location of largest item seen so far.
           for (int j = 1; j <= lastPlace; j++)
          {
              if (A[j] > A[maxLoc])
              {
                    maxLoc = j;
              }
          }
          int temp = A[maxLoc];  // Swap largest item with A[lastPlace].
          A[maxLoc] = A[lastPlace];
          A[lastPlace] = temp;
      }  // end of for loop

}
```

المصفوفات متعددة الأبعاد Multi dimensional Array

فى الفقرات السابقة تم شرح مصفوفة البعد الواحد One Dimension والتى تتكون فى صف واحد أو عمود واحد والتى تأخذ الشكل التالى:

$$A = \begin{bmatrix} 3 \\ 6 \\ 5 \\ 7 \\ 9 \end{bmatrix} \qquad B = \begin{bmatrix} 3 & 6 & 5 & 7 & 9 \end{bmatrix}$$

فى الشكل توجد المصفوفة A التى تتكون من 5 صفوف وعمود واحد والمصفوفة B التى تتكون من 5 أعمدة وصف واحد وكما أشرنا يتم التعامل مع عناصر المصفوفة باسم المصفوفة وترتيب لعنصر، فمثلاً العنصر الأول (3) الموجودة فى المصفوفة A يتم الإشارة إليه بالشكل A [0]، وهكذا، كل ذلك بالنسبة لمصفوفات ذات اتجاه او بعد واحد.

ولكن هناك نوع آخر من المصفوفات الذى يسمى متعدد الأبعاد ويتكون من اكثر من صف واكثر من عمود كما بالشكل:-

C-

3	5	7	9
2	5	9	6
4	3	1	2

فى هذا الشكل توجد مصفوفة بالاسم C عبارة عن 3 صفوف و4 أعمدة لذلك تسمى مصفوفة 4*3 ويتم الإشارة او التعامل مع اى عنصر بتحديد رقم الصف ورقم العمود الذى يقع فيه هذا العنصر فمثلاً العنصر الأول فى المصفوفة 2 وقيمته 3 يشار إليه كما يلى:

```
C [0] [0]
```

وذلك لان يقع فى الصف رقم 0 والعمود 0 فى حين العنصر الثانى فى نفس الصف يشار إليه بالصيغة

```
C [0] [1]
```

لأنة يقع فى الصف 0 فى العمود 1 وهكذا. باقى العناصر.

الإعلان عن مصفوفة متعددة الأبعاد

ويتم ذلك باسم المصفوفة ونوعها وتحديد عدد الصفوف وعدد الأعمدة كما يلى:

```
int XY [ ] [ ] = New int [5] [6];
```

فى هذا السطر تم الإعلان عن مصفوفة بالاسم XY ذات بعدين من النوع int ثم استعمالnew لتحديد عدد الصفوف بالعدد 5 والأعمدة بالعدد 6 ويتم تسجيل او التعامل مع اى عنصر بتحديد الصف والعمود فمثلاً لوضع قيمة فى العنصر الموجود فى الصف الثانى (2) والعمود الثالث (3) نكتب السطر التالى:

```
[2] [3] = 90;
```

وللتعامل مع المصفوفة ذات البعدين تستعمل اثنين تكرار متداخلة باستعمال for loop والذى يسمى Nested كما سيلى فى فقرة التكرار Loops

بلوك الأوامر Block Statements

كما يوجد فى لغة C ما يسمى بلوك كذلك يوجد فى لغة Java وهو عبارة عن اى مجموعه من الجمل او الأوامر محصورة بين الأقواس { } كما بالشكل التالى:

```
{
        :
        :
        :
Statement 1
Statement 2
        :
        :
}
```

فى هذا الشكل يتم حصر جمل Statement 1 و Statement 2 بين الأقواس ليتعامل كبلوك.

ولكن السؤال ما الفائدة وراء إنشاء بلوك جمل ؟

الفائدة هى إمكانية الإعلان عن متغيرات داخل البلوك ليس لها علاقة بما قبل او بعد البلوك مثل تعريف متغير داخل بلوك داخل دالة فهو خاص بها

مثال:

فى هذا المثال نوضح كيفية استعمال التكرار المتداخل مع المصفوفات وكيفية التعامل مع عناصر المصفوفات بدقة اكتب سطور هذا البرنامج كما فى الشكل