المصفوفات في لغة Java Arrays

المصفوفة: هي متحول مركب يستطيع تخزين (في الذاكرة) مجموعة من المعطيات المتجانسة أي التي من نفس النوع. ويُدعى عدد عناصر المصفوفة ببُعدها وهو ثابت (خلال التنفيذ).

- يمكن للمصفوفات أن تكون أحادية أو ثنائية البعد أو حتى من أبعاد أعلى، لكن الأبعاد الأعلى من الثنائية ليست شائعة في التطبيقات العملية.

المصفوفات أحادية البعد: One-Dimensional Arrays

* **التصريح عن مصفوفة:**

نوع عناصر المصفوفة [ ] اسم المصفوفة ; (فراغ)

أو

نوع عناصر المصفوفة [ ] اسم المصفوفة ; (فراغ)

- لا يتم تحديد بعد المصفوفة في التصريح.
- يتم حجز خانات المصفوفة عند الحاجة لاستخدامها، ولا يمكن البدء باستخدام مصفوفة مُصرَّح بها قبل حجز خاناتها.

مثال:

```
int A[];      مصفوفة أعداد صحيحة
char[] c;     مصفوفة محارف
```

* **حجز خانات المصفوفة: (في الذاكرة).**

اسم المصفوفة = new نوع عناصر المصفوفة [البعد] ; (فراغ)

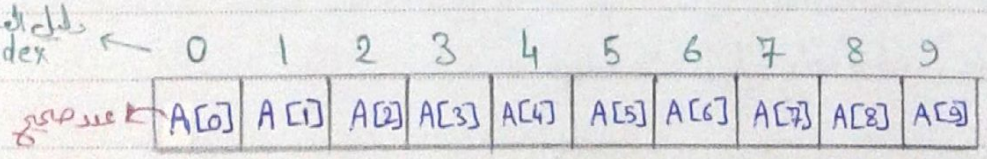
مثال:

```
A = new int [10];
```

- يمكن دمج التصريح وحجز الخانات للمصفوفة في سطر واحد كما يلي:

```
int A[] = new int [10];
```

وهنا يتم حجز 10 أماكن متجاورة في الذاكرة لعشرة متحولات صحيحة:



/ /

الطريقة Length : هذه الطريقة تفيد (تعطي) بعد المتجهة .

مثلاً: في المثال السابق إذا كتبنا A.Length فإن ذلك يعطي العدد 10 .

* قراءة المتجهة: لقراءة المتجهة A السابقة باستخدام صنف الـ Stdin :

```
for (int i=0 ; i<10 ; i++)
  A[i] = Stdin.readInt () ;
```

أو نكتب:

```
for (int i=0 ; i< A.length ; i++)
  A[i] = Stdin.readInt () ;
```

* طباعة المتجهة: لطباعة المتجهة A السابقة نكتب:

```
for (int j=0 ; j<10 ; j++)
  System.out.print (A[j] + " ") ;
```

أو نكتب:

```
for(int j=0 ; j<A.length ; j++)
  System.out.print (A[j] + " ") ;
```

ملاحظة:

- تتيح لغة Java إمكانية حجز خانات المتجهة (عدد عناصرها) عن طريق متحول متغير n على عكس لغة C++ التي تشترط أن يكون هذا المتحول ثابتاً.
- ولكن يجب الانتباه إلى ضرورة إعطاء قيمة لهذا المتحول قبل الحجز، فلا يمكن حجز عدد غير معروف من الخانات. أي (كمثال):

```
int A[] ;
int n ;
n = Stdin.readInt () ;
A = new int [n]
```

ولا يمكن كتابة السطر الأخير إذا حذفنا السطر الذي قبله ولم نعط قيمة محددة لـ n .

- كما أنه لا يمكن تغيير عدد عناصر المتجهة بعد حجزها، أو القيام بالحجز مرتين.

المتغيرات ثنائية البعد (المصفوفات):

المصفوفة: هي عبارة عن متغيرة من المتغيرات، وتكون كل عناصرها متجانسة.

* التصريح عن مصفوفة:

نوع عناصر المصفوفة ـ فراغ ـ اسم المصفوفة [ ] [ ] ;

أو نوع عناصر المصفوفة [ ] [ ] ـ فراغ ـ اسم المصفوفة ;

مثال: للتصريح عن مصفوفة من المحارف نكتب:

```
char  c[][];
char[][]  c;
```

أو

* حجز خانات المصفوفة: (في الذاكرة).

اسم المصفوفة = new ـ فراغ ـ نوع عناصر المصفوفة [n] [m] ;

حيث n يمثل عدد الأسطر و m عدد الأعمدة.

مثال:

```
c = new  char [5] [7];
```

أي أنّ c مصفوفة من البعد 5x7.

- يمكن عند الحجز تحديد عدد الأسطر، وترك عدد الأعمدة غير محدود، والعكس غير صحيح.
- تعطينا الطريقة length عدد أسطر المصفوفة السابقة بالعبارة c.Length.
- وتعطينا عدد أعمدتها بالعبارة C[i].length حيث i ≤ n-1

* قراءة (إدخال) مصفوفة: لإدخال عناصر المصفوفة الحرفية c نكتب:

```
for (int i=0 ; i< c.length ; i++)
    for (int j=0 ; j < c[i].Length ; j++)
        c[i][j] = Stdin.readChar();
```

* طباعة عناصر مصفوفة: لطباعة عناصر المصفوفة السابقة نكتب حلقتي for السابقتين دون أي تغيير ونبدل السطر الأخير بـ:

```
System.out.print(A[i][j]+" ");
```

ملاحظة: هنا تم طباعة كل عناصر المصفوفة بجانب بعضها على سطر واحد وليس على شكل مصفوفة.

تمرين: لم يحلها الدكتور

* اكتب برنامجاً بلغة الجافا يقوم بإدخال عددين طبيعيين وبيان فيما إذا كانا متحابين أم لا.

الحل:

```
class Amicable {
 static void main (String args [])
 { int n,m ;
   do { n= Stdin.readInt();} while (n<0);
   do { m= Stdin.read Int ();} while (m<0);
   int p=0;      int q=0 ;
   for (int i=1 ; i<n ; i++)
     if (n% i==0) p=p+i ;
   for (int i=1 ; i<m ; i++)
     if (m% i==0) q=q+i ;
   if (m==p && n==q) System.out.print("Amicable");
   else System.out.print ("Not amicable");
} }
```

* اكتب برنامجاً يقوم بإدخال العدد الطبيعي n وطباعة جميع الأعداد التامة المحصورة في المجال [0, n].

الحل:

```
class Perfect {
static void main (String args [])
{ int n;        int p=0;
  do {n= Stdin.readInt();} while (n<0);
  for (int i=1 ; i<=n ; i++)
  { for ( int j=1 ; j<i ; j++)
      if (i% j==0) p=p+j;
    if(p==i) System.out .println (i);
    p=0 ;
  } } }
```

/ /

* اكتب برنامجاً يقوم بإدخال متجهة أعداد صحيحة من البعد n ، حيث n عدد طبيعي مُدخل ، ثم ينجز مايلي :

١- إيجاد أصغر عنصر في المتجهة المدخلة وطباعته مع دليله .
٢- حساب وطباعة المتوسط الحسابي لعناصر هذه المتجهة .
٣- حساب وطباعة جداء وكميات العناصر ذات الأدلة الفردية في المتجهة .
٤- حساب وطباعة مجموع مربعات العناصر الزوجية في المتجهة .
٥- بين فيما إذا كانت المتجهة المدخلة قابلة للطي أم لا .
٦- ترتيب عناصر المتجهة ترتيباً تصاعدياً وطباعتها .
٧- طباعة عناصر المتجهة مرتبة ترتيباً تنازلياً .

الحل :

```
class Array1 {
static void main (String args[])
{ int A[];
  System.out.print ("n=");
  int n = Stdin.readInt();
  while (n<1) {
      System.out.println ("n must be more than 1");
      System.out.print ("n=");
      n= Stdin.readInt();
    }
  A= new int [n];

  for (int i=0; i<n ; i++)
   { System.out.print ("A[" + i + "]=");
     A[i]= Stdin.readInt();
   }
```

/ /

حل الطلب الأول:

```
int   x = A[0];
for (int  i=1 ; i<n ; i++)
    if (A[i] < x)   x = A[i];
System.out.println ("The Min:");
for (int i=0 ; i<n ; i++)
    if (A[i] == x)
        System.out.println ("A[" + i + "]=" + x);
```

حل الطلب الثاني:

```
x = 0;
for (int i=1 ; i<n ; i++)
    x = x + A[i];
System.out.println ("Average=" + (float) x /n);
```

ترقية

حل الطلب الثالث:

```
x = 1;
for (int i=1 ; i<n ; i=i+2
    x = x* A[i]* A[i]* A[i];
System.out.println (x);
```

حل الطلب الرابع:

```
x = 0;
for (int i=0 ; i<n ; i++)
    if (A[i] % 2 == 0)   x = x + (A[i]* A[i]);
System.out.println (x);
```

/ /

حل الطلب الخامس:

```
boolean t=true;
for (int i=0 ; i< n/2 ; i++)
   if (A[i]!= A[n-1-i])  t=false;
if (t)    System.out.println(" قابلة للطي ");
else      System.out.println(" غير قابلة للطي ");
```

حل الطلب السادس:

```
int tmp;
for (int i=0 ; i<n-1 ; i++)
   for( int j=i+1 ; j<n ; j++)
      if (A[i]> A[j]) { tmp = A[i];
                        A[i] = A[j];
                        A[j] = tmp;
                      }

for (int i=0 ; i<n ; i++)
   System.out.print (A[i] + " ");
System.out.println();
```

حل الطلب السابع:

```
for (int i=n-1 ; i>=0 ; i--)
   System.out.print (A[i] + " ");

} // end of main
} // end of class
```