بنام خدا

سری های آموزشی آشنایی با زبان برنامه نویسی C++

قسمت ششم : آرایه ها

ویرایش : 1

یادآوری از قسمت 1 :

آرایه های یک بعدی : به طور کلی آرایه ها نوعی پیشرفته تر از متغیر ها هستند که می توانند چندین مقدار را در خود ذخیره کنند .

ساختار کلی آرایه ها به صورت زیر است :

```
array type      array name [number of elements] ;
```

مثلا برای آرایه ی یک بعدی A که دارای n خانه است خواهیم داشت :

A[0]	A[1]	A[3]	...	A[n-1]

نکته ی 1 : در زبان C++ شماره ی خانه ی آرایه ها از 0 شروع می شود تا n-1 .

مثال :مثلا در زیر آرایه ی Doste دارای 5 عنصر است و همچنین از نوع کاراکتری است :

```
int main()
   {
   char Doste[5];
   return 0;
   }
```

Doste[0] Doste[1] Doste[2] Doste[3] Doste[4]

برای مقدار دهی به یک آرایه یک برنامه ی ساده و معمولا تکراری در برنامه ها خواهیم داشت . پس به آن خوب توجه کنید :

```
#include <iostream.h>
int main()
   {
   int a[4];
   cout<<"enter content of 4 elements : "<<endl ;
       for(int i=0; i<4; i++)
            {
            cout<<"enter content of a["<<i<<"]"<<endl;
            cin>>a[i];
            }
   return 0;
   }
```

همچنین برای چاپ آن هم برنامه ای شبیه آن خواهیم داشت :

```
#include <iostream.h>
   int main()
   {
      int a[4];
      cout<<"enter content of 4 elements : "<<endl ;
       for(int i=0; i<4; i++)
            {
            cout<<"enter content of a["<<i<<"]"<<endl;
            cin>>a[i];
            }
       for(i=0; i<4; i++)
            cout<<"content of a["<<i<<"]="<<a[i]<<endl;
   return 0;
   }
```

نکته 2 : در هنگام معرفی یک آرایه مثلا آرایه ی a مقداری که به آن به عنوان تعداد خانه ها می دهیم ، باید مقداری ثابت باشد .
یعنی اگر مانند زیر بنویسیم ، اشتباه است و برنامه خطا خواهد داد :

```
int b=4;
int a[b];
```

چون که متغیر مانند b ، ممکن است در طول برنامه مقدارش عوض شود .

بنابرنین تنها دو راه وجود دارد :
1- اینکه مانند برنامه های قبل عدد بنویسیم . مثلا :

```
int a[4];
```

2- استفاده از نوع داده ی const int که مقدار آن همواره ی مقداری است ، ثابت . مثلا :

```
const int b=4;
int a[b];
```

شاید در بعضی از برنامه ها این مشکل بوجود آید که ما نمی دانیم تعداد خانه های یک آرایه باید به چه تعداد باشد ؟
در اینصورت بهترین راه حل دادن مقداری بزرگ در برابر مبحث مورد نظر است . مثلا عدد 512 در این زمینه بسیار پر کاربرد است .

در c++ این امکان وجود دارد که بتوانید خانه های آرایه ای را از قبل در برنامه مقدار دهی کنید . مثلا برای یک آرایه یک بعدی 5 خانه ای می خواهیم مقداری دهی پیش فرض انجام دهیم :

```
int a[5]={24,33,49,55,63};
```

همچنین می توانیم این کار را در قسمت کد برنامه نیز انجام دهیم :

```
#include <iostream.h>
int main()
   {
      int a[5]={24,33,49,55,63};

       for(int i=0; i<4; i++)
            cout<<"content of a["<<i<<"]="<<a[i]<<endl;
   return 0;
   }
```

مثال : برنامه ای بنویسید که مقادیر یک آرایه ی یک بعدی 5 خانه ای را از ورودی گرفته و مجموع مقادیر خانه ها را چاپ کند .

```
#include <iostream.h>
 int main()
       {
       int a[5],sum=0;
            for(int i=0; i<5; i++)
              {
              cout<<"a["<<i<<"]=";
              cin>>a[i];
              sum=sum+a[i];
              }
       cout<<"The sum is " <<sum;
       return 0;
       }
```

- استفاده از آرایه به عنوان آرگومان یک تابع :
آرایه ها را نیز همچون سایر نوع داده ها می توان به یک تابع ارسال کرد. برای اینکار ابتدا باید تابع را بگونه ای تعریف کنیم که یک پارامتر از نوع آرایه را دریافت کند.
ساختار استاندارد استفاده از آرایه به عنوان آرگومان یک تابع به صورت زیر می باشد : (در اینجا مثال برای آرایه ای از نوع int می باشد)
(خوب به آن توجه کنید و آنرا به خاطر بسپارید !)

```
#include <iostream.h>
Void fun(int array[])
  {
  ...
  }


int main()
   {
   int a[10];

   ...

   fun(a);

   ...

   return 0;
   }
```

برای ارسال آرایه موردنظر کافی است که تنها نام آرایه را بدون کروشه استفاده نماییم.(البته لازم می شود اندازه واقعی آرایه را نیز بعدا بعنوان دومین آرگومان به تابع ارسال کنیم.)

توجه کنید که بر عکس انواع داده های دیگر ، هر عملی که به صورت فوق در برنامه به عنوان آرگومان یک تابع استفاده شود ، هر تغییری که در آن توسط تابع رخ دهد در آن ذخیره خواهد شد . (می توانید این مورد را با برنامه های شماره ی قبل در مورد توابع بررسی کنید. این مبحث به تفاوت های بین مقادیر عادی و مقادیر ارجاعی مربوط می شود ؛ که بعدا مفصلا در مورد آن بحث خواهد شد)

بنابرین ، در صورتی که بخواهید آرایه ای را به تابعی طوری بفرستید که در آن هیچ تغییری امکان پذیر نباشد ، آن را از نوع const int تعریف کنید . در اینصورت ، کامپایلر اجازه ی هیچ گونه تغییری را در آن نخواهد داد .

مثال : برنامه ای بنویسید که مقادیر یک آرایه ی یک بعدی 5 خانه ای را از ورودی گرفته و به تابعی ارسال کند . تابع مجموع مقادیر خانه ها را حساب کرده و مقدار آن را به برنامه اصلی بازخواهد گرداند و بعد ما مقادیر آن را در برنامه اصلی چاپ خواهیم کرد.

```
#include <iostream.h>
int fun(int a[])
  {
  int sum=0;
       for(int i=0; i<5; i++)
            {
            sum+=a[i];
            }
  return sum;
  }

 int main()
       {
       int a[5];
            for(int i=0; i<5; i++)
              {
              cout<<"a["<<i<<"]=";
              cin>>a[i];
              }
       cout<<"The sum is " <<fun(a);
       return 0;
       }
```

مثال : برنامه ای بنویسید که اشتراک دو مجموعه را در خروجی چاپ کند :

```
#include <iostream.h>
int main()
   {
  int a[5]={5,4,3,2,1}
        ,b[3]={6,3,1};
  cout<<"Union part is :    " ;
  for(int i=0; i<5; i++)
       {
       for(int k=0; k<3; k++)
            {
             if (a[i]==b[k])
                  cout<<a[i]<<"   ";
            }
       }
   return 0;
   }
```

***نکته بسیار مهم :خروجی یک تابع نمی تواند یک آرایه باشد. برای بازگرداندن یک آرایه از تابع، باید آن را بصورت یک پارامتر خروجی به تابع ارسال نمود.

آرایه های چند بعدی :

آرایه های چند بعدی ، نوع پیشرفته تری از آرایه ها هستند که می توانند اطلاعات بیشتری را در خود ذخیره کنند . و در عوض نیز کار های پیشرفته تری را انجام دهند .

ساختار کلی معرفی این آرایه ها به اینصورت است :

```
type  array-name  [element-size 1][ element-size  2]  ...  [ element-size n] ;
```

بعنوان مثال، اعلان زير يك آرايه دوبعدي را معرفي مي نمايد:

```
int A[2][5] ;
```

این آرایه ی دوبعدی را می توانیم به صورت یک مستطیل بر روی سطح (دوبعد) نشان دهیم :

A[0][0]	A[0][1]	A[0][2]	A[0][3]	A[0][4]
A[1][0]	A[1][1]	A[1][2]	A[1][3]	A[1][4]

که در مستطیل بالا ، آدرس هر خانه را هم می بینید .

در واقع در یک آرایه ی دو بعدی داریم :

A[a][b] : a=تعداد ستون , b= تعداد سطر

توجه : در زبان C بر خلاف زبان های دیگر از جمله پاسکال و بیسیک ، مقدار اول ستون و مقدار دوم سطر می باشد.

همچنین آرایه ی سه بعدی هم به همین صورت :

A[a][b][c] : a=تعداد عرض , b= تعداد ستون , c=تعداد سطر

که شکلی به صورت یک مکعب مستطیل خواهد شد .

حال آیا می توانید حدس بزنید آرایه ی چهار بعدی به چه شکل خواهد بود ؟!! (اگر فهمیدید به من هم بگویید !)

نکته : مانند آرایه های یک بعدی ، اندیس سطرها و ستونها هر دو از 0 آغاز مي گردند.

همچنین مانند آرایه های یک بعدی برای مقدار اولیه دادن به آرایه های دو و سه بعدی به اینصورت عمل خواهیم کرد :

مثلا :

```
int A[3][4] = { {12, 5, 3, 8} , {-3, 7, -9, 2}, {4, 22, 18, 6} };
```

که همان این شکل است :

12	5	3	8
-3	7	-9	2
4	22	18	6

یا مثلا برای آرایه ی سه بعدی خواهیم داشت :

```
int A[2][3][4] = { { {12, 5, 3, 8} , {-3, 7, -9, 2}, {4, 22, 18, 6} } , { {8, 1, -3, 4} , {-2, 8, 11, 21} , {7, 3, -15, -8} } };
```

ارسال آرایه های چند بعدی به توابع :

ارسال آرایه های چند بعدی تقریبا شبیه ارسال آرایه های یک بعد است .

فقط یک نکته را به خاطر داشته باشید که :

شاید تصور کنید که برای تعریف یک آرایه دوبعدی بعنوان پارامتری از یک تابع، تنها قرار دادن دو علامت [] کافی است و نیازی به ذکر ابعاد آن نیست. اما اینگونه نیست، بلکه برنامه نویس باید تعداد ستونهای آرایه دوبعدی را صریحا مشخص نماید، اما نیازی به تعیین تعداد ردیفهای آن و ... نیست. بعنوان مثال فرض کنید تابعی مانند test داریم که بعنوان ورودی یک آرایه دو بعدی و تعدادی پارامتر دیگر دریافت می کند.

مثلا تعریف تابع بصورت زیر اشتباه است:

```
Int  test(int A[ ][ ]...)
 {
...
 }
```

اما کد زیر تعریفی درست است :

```
Int  test(int A[ ][4]...)
 {
...
 }
```

مسئله : برنامه ای بنویسید که حاصل جمع دوماتریس زیر را در خروجی چاپ کند .

-3	-7
4	34
45	7

و

-4	8
53	-17
4	4

```
#include <iostream.h>
int main()
   {
  int a[3][2]={{-3,-7},{4,34},{45,7}}
        ,b[3][2]={{-4,8},{53,-17},{4,4}}
        ,c[3][2];

       for(int i=0; i<3; i++)
            {
            for(int k=0; k<2; k++)
                  {
                  c[i][k]=a[i][k]+b[i][k];
                  cout<<c[i][k]<<"   ";
                  }
            cout<<endl;
            }
   return 0;
   }
```

می توانید برنامه ی جمع ماتریس ها را در حالت کلی برای دو ماتریس m*n خانه ای بازنویسی کنید .

مثال : برنامه ای بنویسید که حاصلضرب دو ماتریس زیر را بدست آورد :

-3	-7
4	34
45	7

×

-4	23	13	8
53	11	-1	-17

```
#include <iostream.h>
int main()
   {
  int a[3][2]={{-3,-7},{3,34},{45,7}}
        ,b[2][4]={{-4,23,13,8},{53,11,-1,-17}}
        ,c[3][4];
  int s;
  for (int i=0; i<3; i++)
             {
             for(int j=0; j<4; j++)
                    {
                    s=0;
                    for(int k=0; k<2; k++)
                           s+=a[i][k]*b[k][j];
                    c[i][j]=s;
                    }
             }
  for(i=0; i<3; i++)
        {
        for(int j=0; j<4; j++)
             cout<<c[i][j]<<"      " ;
        cout<<endl;
        }
   return 0;
   }
```

*** می توانید این برنامه را برای ضرب یک ماتریس m*k در ماتریس k*n که در نهایت ماتریسی بصورت m*n خواهد شد بازنویسی کنید .

تمرینات :

1- برنامه ای بنویسید که دترمینان یک ماتریس n*n را محاسبه کند .
2- برنامه ای که طول و مقادیر یک آرایه ی یک بعدی را گرفته و سپس از ورودی یک مقدار بگیرد و در خروجی چاپ کند که آیا این مقدار در آرایه ای که از ورودی گرفته وجود دارد یا نه ؟!

پایان قسمت ششم!

نویسنده : دانیال خشابی
ویرایش و صحت مطالب : نوید مردوخ روحانی

www.mrh.ir
www.majidonline.com