آرایه ها

ضرورت وجود آرایه ها

- **مثال: برنامه ای بنویسید که ۵ عدد را بگیرد و میانگین و واریانس آنها را محاسبه کند.**

نکته: میانگین و واریانس n نقطه X_0، X_1و....وX_{n-1} از روابط زیر به دست می آیند:

$$\mu = \frac{\sum_{i=0}^{n-1} x_i}{n} \qquad \sigma^2 = \frac{\sum_{i=0}^{n-1} (x_i - \mu)^2}{n}$$

اگر تنها هدف به دست آوردن میانگین بود به راحتی می توانستیم آن را انجام دهیم:

```
float x;
sum=0;
for (i=0;i<5;++i)
{
    scanf("%f",&x);
    sum+=x;
}
ave=sum/5;
```

ادامه

- **در اینجا برای به دست آوردن واریانس نیاز به همه x_i ها داریم.**

```
float x0,x1,x2,x3,x4;
float ave,var;
scanf("%f%f%f%f%f",&x0,&x1,&x2,&x3,&x4);
ave=(x0+x1+x2+x3+x4)/5;
var=(pow(x0-ave,2)+pow(x1-ave,2)+pow(x2-
   ave,2)+pow(x3-ave,2)+pow(x4-ave,2))/5;
```

ادامه

- **به این ترتیب کد نامناسبی خواهیم داشت. این موضوع به ازای داده های زیادتر (مثلا ۱۰۰۰ عدد) بیشتر نمود پیدا می کند.**
- **با تعریف آرایه ها می توان چنین برنامه هایی را خیلی ساده و خلاصه پیاده کرد.**
- **به طور کلی آرایه ها در مواردی به کار برده می شوند که با مجموعه ای از داده های *هم‌نوع* سروکار داشته باشیم.**

تعریف آرایه

- **یادآوری: با دستور تعریف متغیر مثل** int x; **مکانی در حافظه به نام** x **به برنامه تخصیص داده می شد.**
- **در زبان** C **اگر بنویسیم** int x[5]; **، پنج مکان مجاور هم در حافظه به برنامه تخصیص داده می شود. این مکان ها به صورت** x[0]، x[1]، x[2] **و** x[4] **نامیده می شوند.**
- **اصطلاحا می گوییم آرایه ای به نام** x **، از نوع** int **و با ۵ عنصر تعریف کرده ایم.**
- **به** 0، 1، 2، 3 **و** 4 **اصطلاحا اندیس آرایه گفته می شود.**
- **به همین ترتیب می توان آرایه هایی ازانواع دیگر مثل** float، char، short **و ....داشت.**

مقدار دهی به آرایه ها

- **سه روش برای مقدار دهی به آرایه ها وجود دارد:**
  - **با دستورات ورودی مثل** scanf **یا** getche**:**

```
ch[0]=getche();
scanf("%d",&x[2]);
```

  - **مقدار دهی مستقیم:**

```
ch[0]='a';
x[2]=23;
```

  - **مقداردهی هنگام تعریف: (صفحه بعد)**

مقداردهی هنگام تعریف

- **می توان آرایه ها را (تنها)هنگام تعریف به صورت زیر مقداردهی کرد:**

```
int x[3]={2,3,-5};
```

در اینجا ۲ در x[0]**، ۳ در** x[1] **و ۵– در** x[3] **قرار می گیرد.**

مثال های دیگر:

```
float f[2]={0.34,4.33};
char ch[4]={'a','d','x','v'};
```

چند سوال

- **اگر در مقداردهی هنگام تعریف، تعداد مقادیر نوشته شده داخل { } از اندازه مشخص شده برای آرایه بیشتر باشد خطا اعلام می شود.**

```
int x[2]={3,4,6,7,8}
```

- **اگر تعداد مقادیر کمتر باشد، بقیه صفر در نظر گرفته می شوند:**

```
int x[3]={3,4}
```

در اینجا x[2]=0 **می شود.**

مثال دیگر:

```
int x[100]={0}
```

همه مقادیر صفر می شوند. (این روشی برای صفر کردن همه عناصر آرایه است).

- **اگر بنویسیم:** int x[ ]={2,3} **اندازه آرایه ۲ در نظر گرفته می شود.**

یک نکته

Int i=8;

Int x[i];

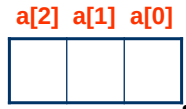
اندازه آرایه باید یک عدد ثابت باشد. بنابراین نوشتن به صورت بالا نادرست است.

ادامه آرایه

- **تعریف آرایه:**

int a[3];

- **آرایه ای به نام a از نوع int و با اندازه ۳ عنصر تعریف کرده ایم.**



- **به عناصر آرایه می توان از طریق اندیسشان دسترسی داشت.**

```
a[0]=4;
scanf("%d",a[1]);
for (i=0;i<3;++i)
    scanf("%d",&a[i]);
```

ادامه آرایه

مقداردهی اولیه:

```
float a[4]={4.34,9.45,3.23,1.02};
```

اگر تعداد مقادیر از اندازه مشخص شده برای آرایه بیشتر باشد خطا رخ می دهد:

```
float a[3]={7.34,1.2,3.4,1.44};
```

اگر تعداد مقادیر از اندازه کمتر باشد، بقیه مقادیر صفر در نظر گرفته می شوند:

```
float a[3]={2.34,3.23};
```

ادامه

مثال) محاسبه میانگین و واریانس ۵ عدد

```
#include <stdio.h>
#include <math.h>
void main( )
{
  float x[5];
  float sum=0 , ave;
  int I;
  for (i=0;i<5;++i)
  {
     scanf("%f",&x[i]);
     sum+=x[i];
  }
ave=sum/5;
var=0;
for (i=0;i<5;++i)
  var+=pow (x[i]-ave,2)/5;
printf("The average=%f",ave);
Printf("The variance=%f",var)
}
```

مثال

```
#include <stdio.h>
void main( )
{
  int i;
  int x[4];
  for (i=0;i<4;++i)
      scanf(“%f”,&x[i]);
  for (i=0;i<4;++i)
     printf(“\t%f”,x[i]);
}
```

حلقه اول مقادیر را دریافت می کند و در آرایه x قرار می دهد و حلقه دوم این مقادیر را چاپ می کند.

مثال) برنامه ای بنویسید که ۴۰ مقدار را از کاربر دریافت کرده و آنها در یک آرایه بریزد و سپس ماکزیمم مقادیر و اندیس عنصر ماکزیمم را محاسبه کند.

```
#include <stdio.h>
void main()
{   int i;
    float x[۴0];
    float max;
    int max_index;
    for (i=0;i<۴0;++i)
    {         printf("Enter  number %dth:",(i+1));
          scanf("%f",&x[i]);
    }
    max=x[0];
    max_index=0;
    for (i=1;i<۴0;++i)
          if (x[i]>max)
          {
                    max=x[i];
                    max_index=i;
          }
    printf("\nThe maximum=%f",max);
    printf("\nAnd its index=%d",max_index);
}
```

مثال ۵) برنامه ای بنویسید که ۱۰ عدد از کاربر بگیرد آنها را به صورت صعودی مرتب کرده و چاپ کند.

برنامه از دو بخش تشکیل شده است:

دریافت اعداد

مرتب کردن آنها

1. **دریافت اعداد را می توان با تعریف یک آرایه و نوشتن قطعه برنامه ای به صورت زیر انجام داد:**

```
float x[10];
int i;
for (i=0;i<10;++i)
scanf("%f",&x[i]);
```

ادامه

۲. به مسئله مرتب سازی اصطلاحا sorting **گفته می شود.**
برای مرتب سازی روش های متعددی وجود دارد. مانند :
bubble sort**(مرتب سازی حبابی)**، quick sort، merge sort **و.....**

در این مثال از bubble sort **که یکی از ساده ترین آنهاست استفاده می شود.**

برای مرتب سازی صعودیBubble sort

- ابتدا دو عنصر اول و دوم آرایه با هم مقایسه می شوند.اگر عنصر اول از دوم بزرگتر بود جای دو عنصر عوض می شود. سپس عناصر دوم و سوم مقایسه می شوند و مشابه قبل تعویض مکان در صورت لزوم انجام می گیرد. همین کارها برای عناصر ۳ و ۴، ۴ و ۵، و....انجام می شود تا به انتهای آرایه برسیم.
- زمانی که به انتهای آرایه برسیم ماکزیمم مقادیر در آخرین مکان آرایه قرار گرفته است و به عبارت دیگر

زمانی که یک باراز ابتدا تا انتهای آرایه پیمایش شود عنصر آخر می شود.

- در مرحله بعد این پیمایش و اعمال گفته شده روی عناصر اول تا یکی مانده به آخر انجام می شود و در نتیجه :

با دو بار پیمایش ۲ عنصر مرتب می شوند.

- در مرحله بعد پیمایش از ابتدا تا عنصر دوتا مانده به آخر آرایه انجام می شود و در نتیجه: با سه بار پیمایش سه عنصر مرتب می شود.
- به این ترتیب اگر آرایه ، n عنصری باشد با n-1 پیمایش ، کل آرایه مرتب می شود.

تعویض دو مقدار

- فرض کنید دو متغیر x=23 و y=34 داشته باشیم و بخواهیم مقدار آنها را با هم تعویض کنیم.
- با تعریف متغیری به نام temp و نوشتن قطعه کد زیر می توان این کار را انجام داد:

```
float temp;
temp=x;
x=y;
y=temp;
```

برنامه مثال ۵

```
#include <stdio.h>
void main()
int i,j;    {
float A[10];
float temp;
for (i=0;i<10;++i)
scanf("%f",&A[i]);
for (i=0;i<9;++i)
for (j=0;j<9-i;++j)
if (A[j]>A[j+1])
{
temp=A[j];
A[j]=A[j+1];
A[j+1]=temp;
}
for (i=0;i<10;++i)
printf("\n%f",A[i]);
}
```

آرایه ها به عنوان ورودی و برگشتی توابع

- آرایه ها مانند سایر متغیرها می توانند ورودی یا برگشتی تابع باشند.

آرایه به عنوان ورودی تابع

مثال ۶: تابعی بنویسید که یک آرایه را بگیرد و ماکزیمم آن را برگرداند:
ورودی: آرایه برگشتی: یک عدد

```
float max(float x[10])
{
    int i;
    float  max=x[0];
    for (i=1;i<10;++i)
    if (x[i]>max)
max=x[i];
    return max;
}
```

برنامه کامل مثال ۶

```
#include <stdio.h>
float max(float x[])

void main()
{
    int i;
    float A[10];
    for (i=0;i<10;++i)
scanf("%f",&A[i]);
printf("\nThe maximum=%f",max(A));
}

{
    int i;
    float  max=x[0];
    for (i=1;i<10;++i)
    if (x[i]>max)
max=x[i];
    return max;
}
```

نحوه تعریف رشته ها

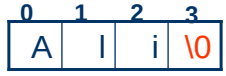
- **در C، رشته ها به صورت آرایه ای از کاراکترها پیاده می شوند.**
  - **با نوشتن** char st [20]; **می توانیم رشته ای به نام** st **با ماکزیمم طول ۲۰ تعریف کرده ایم.**

نحوه مقداردهی

- **نحوه مقداردهی: به دو صورت می توان به یک رشته مقدار داد:**
  1. **با استفاده از دستورات ورودی: مثل** scanf **با استفاده از کارکتر کنترلی** %s
     - scanf("%s",st) **یا** scanf("%s",&st);
  2. **به صورت مستقیم هنگام تعریف رشته**

     char st[20]="Ali";
- **نحوه قرار گرفتن رشته در حافظه به این صورت است:**

- **C در انتهای هر رشته به طور خودکار کاراکتر NULL (پوچ–تهی) (که با علامت \0 نشان داده می شود) را قرار می دهد.**
- **در نتیجه طول واقعی یک رشته یکی بیشتر از تعداد کاراکترهایش است.**



ادامه

نکته:کد اسکی کاراکتر NULL **، صفر است.**
printf(“%c”,’\0’);**کاری انجام نمی شود .**
printf(“%d”,’\0’);**صفر چاپ می شود.**

نحوه چاپ یک رشته: با دستور printf **و استفاده از کاراکتر کنترلی** %s

ادامه

- **مثال ۱)**

```
#include <stdio.h>
void main()
{
char st[20]="Hello";
printf("%s",s);
}
```

رشته ای به نام st **با مقدار** Hello **تعریف شده .خروجی این برنامه پیغام** Hello **است.**
می توان این برنامه را با یک دستور زیر نوشت:

```
printf(“%s”,”Hello”);
```

ادامه

مثال ۲) رشته ای را از کاربر دریافت کرده و آن را چاپ می کند.

```
#include <stdio.h>
void main()
{
    char ch[4];
scanf("%s",ch);
printf("%s",ch);
}
```

ادامه

مثال ۳)

```
#include <stdio.h>
void main()
{
  char str[23]="Hello";
   printf("%c",str[2]);
}
```

با استفاده از اندیس می توان به کاراکترهای داخل رشته دسترسی داشت.

ادامه

نکته: چند وضعیت مختلف در مقداردهی هنگام تعریف
۱. رشته از اندازه آرایه کوچکتر است.
char s[8]="abc";
بقیه عناصر را NULL **در نظر می گیرد.**
۲. رشته از اندازه آرایه بزرگتر است
char s[3]="aghugii";
خطا
۳. رشته مساوی اندازه آرایه است.
char s[3]="ali";
خطا نیست. در اینجا کاراکتر NULL **جزو رشته حساب نمی شود.**

ادامه

مثال ۴) برنامه ای بنویسید که طول یک رشته را حساب کند (کاراکتر NULL در طول رشته در نظر گرفته نشود).

```
#include <stdio.h>
#Include <string.h>
int strlen(char str[20])
{
    int i=0;
while (str[i] != '\0')
    i++;
    return i;
}
void main()
{
    char str[20];
    scanf("%s",str);
printf("%d",strlen(str));
}
```

ادامه

- مثال ۴) برنامه ای بنویسید که یک رشته از کاربر بگیرد و آن را به رشته ای که تمام حروفش بزرگ است تبدیل کند. (مثلا "abAF" به "ABAF" تبدیل شود.

```
#include <stdio.h>
void main()
{
    char str[20];
    int i;
printf("Enter a string:");
    scanf("%s",str);
    i=0;
while (str[i] != '\0')
    {
        if (str[i]>=97 && str[i]<=122)
printf("%c",str[i++]-32);
    }
}
```

چند تابع کتابخانه ای برای کار با رشته ها

strlen: ورودی آن یک رشته و برگشتی آن طول رشته است.

```
x=strlen("ali");
printf("%d",x);
```

puts: برای چاپ رشته:

```
puts("ali is a student");
char ch[20]="ali;
puts(ch);
```

عملکردی مشابه printf دارد؛ با دو تفاوت:

- فقط برای نمایش رشته به کار می رود و نمی تواند مقدار متغیر ها را نشان دهد.
- بعد از چاپ رشته به خط بعد می رود.

gets: برای دریافت رشته از کاربر

```
char ss[20];
gets(ss);
```

ادامه

strcat : برای الحاق دو رشته به یکدیگر به کار می رود:(strcat(s1,s2 رشته s2 را به انتهای s1 اضافه می کند.

```
#include <string.h>
#include <stdio.h>
void main()
{
char s1[20]="Ali";
char s2[20]="Mehdi";
strcat(s1,s2);
puts(s1);
puts(s2);
}
```