

آرایه

یکی از پرکاربردترین ساختارهای داده آرایه‌هاست که اغلب برای پیاده‌سازی داده‌های انتزاعی خطی بکار می‌رود.

[تعریف آرایه](#)
[آرایه‌های یک بعدی](#)
[آرایه‌های دو بعدی](#)
[محیط به فضا و آدرس](#)
[آرایه‌های پویا](#)
[الگوریتم‌های درج و حذف](#)

تعریف آرایه

آرایه (array) لیست متناهی از عناصر داده‌ای هم‌نوع است.

محل یک عنصر درون آرایه توسط اندیس (Index) معین می‌شود. عنصر a_i در مکان i ام آرایه قرار می‌گیرد به این ترتیب می‌توان به صورت تصادفی مقدار آرایه را بازیابی کرد. البته با روش ترتیبی هم می‌توان به مقادیر عناصر لیست دسترسی پیدا کرد.

به عناصر آرایه ممکن است به کمک مجموعه‌ای از اندیس‌ها مراجعه شود.

فرم آلی تعریف آرایه در زبان برنامه‌نویسی Pascal به صورت زیر است:

ArrayName : ARRAY [IndexType1, IndexType2, ..., IndexTypen] OF Type;

IndexType مجموعه اندیس‌ها را تعیین می‌کند و می‌تواند از هر نوع ممکن باشد. اگر $\{R^+\}$ فعال نباشد کنترلی روی اندیس‌های غیر مجاز نیست. اگر فعال باشد در صورت بی‌تفاه از اندیس غیر مجاز پیغام خطای Range Check Error صادر می‌شود.

در زبان برنامه‌نویسی C آرایه به صورت کلی زیر تعریف می‌شود:

Type ArrayName[Size1] [Size2] ... [Sizen];

Size تعداد عناصر یک بعد آرایه را تعیین می‌کند. در زبان C کلیه اندیس‌ها عددی هستند و از صفر شروع می‌شوند (0-based indexing) و کنترلی روی محدوده اندیس‌ها وجود ندارد.

در اکثر زبان‌ها به آرایه به صورت ایستا حافظه اختصاص می‌دهند و اندازه آرایه در طول اجرای برنامه ثابت است. مگر اینکه حافظه پویا صریحاً توسط برنامه‌نویس بی‌تفاه شود.

آرایه‌های یک بعدی

آرایه یک بعدی مجموعه متناهی از زوج‌ها به صورت $\langle \text{اندیس}, \text{مقدار} \rangle$ است. بدین معنی که، به ازای یک اندیس یک مقدار مربوط به آن وجود دارد.

برای تعریف آرایه یک بعدی یک مجموعه اندیس تعریف می‌شود.

مثال (C). آرایه num با ۲۰ عنصر صحیح تعریف شده است. عناصر آرایه در خانه‌های $\text{num}[0], \text{num}[1], \dots, \text{num}[19]$ ذخیره می‌شوند.

int num [20];

مثال (C). آرایه کاراکتری num با ۴ عنصر تعریف و مقداردهی شده است.

```
Char num[4]={'d','a','t','a'};
```

یا

```
Char num[4]="data";
```

آرایه های دوبعدی

یک آرایه دو بعدی مجموعه ای با $m \times n$ عنصر داده ای است که هر عنصر آن با یک جفت اندیس مشخص می شود.

آرایه دو بعدی را می توان به جدولی تشبیه کرد که دارای m سطر و n ستون است. هر سطر شامل عناصری است که آندیس مای اول آنها برابر است و هر ستون شامل عناصری هستند که آندیس مای دوم آنها برابر هستند.

آرایه مای دوبعدی به عنوان ماتریس به آمار می روند.

در تعریف آرایه دو بعدی دو مجموعه اندیس معین می شود. اندیس اول تعداد سطرها و اندیس آرایه تعداد ستونها را مشخص می کند.

مثال (Pascal). آرایه Table از نوع اعداد حقیقی با ۴ سطر و ۳ ستون تعریف شده است.

```
Table : Array[1..5,3..6] of Real;
```

یا

```
Table : Array[1..5] of Array [3..6] of Real;
```

مثال (C). آرایه A از نوع اعداد صحیح با ۳ سطر و ۴ ستون

```
int A[3][4];
```

مثال (C). آرایه دوبعدی num را می توان به دو صورت مقداردهی اولیه کرد.

```
int num [4][3]={ {15,6,13},{9,17,2},{4,5,4},{10,11,12}} ;
```

یا

```
int num [4][3]={15,9,13,9,17,2,4,5,4,10,11,12};
```

آرایه های چندبعدی

آرایه n بعدی مجموعه ای از $m_1 \times m_2 \times \dots \times m_n$ عنصر داده ای است که هر عنصر توسط n اندیس نظیر i_1, i_2, \dots, i_n مشخص می شوند. آرایه مای چند بعدی در حافظه به صورت دنباله ای از خانه های پشت سر هم ذخیره می شوند.

محاسبه فضا و آدرس

هر متغیری که تعریف می شود مقداری فضای حافظه را به خود اختصاص می دهد که به نوع داده متغیر بستگی دارد. برای بدست آوردن میزان فضای اشغال شده وسیله یک آرایه آفای است که تعداد عناصر آرایه در تعداد بایت مای نوع آرایه ضرب شود. تعداد عناصر آرایه را طول یا اندازه آرایه می گویند.

در زبان Pascal اندازه آرایه به صورت زیر محاسبه می شود :

$A[L_1..U_1, L_2..U_2, \dots, L_n..U_n]$

$$\prod_{i=1}^n U_i - L_i + 1$$

در زبان C اندازه آرایه به صورت زیر محاسبه می شود:

$A[M1][M2] \dots [Mn]$

$$\prod_{i=1}^n M_i$$

مثال (Pascal). آرایه زیر ۲۴ عنصر کاراکتری دارد.

A: Array [1..2, 1..3, 1..2, 1..2] of Char;
 $(2-1+1)(3-1+1)(2-1+1)(2-1+1) = 24$.

مثال (C). میزان فضای اشغال شده توسط آرایه num به صورت زیر محاسبه می شود:

```
int num [10][5] ;
10×5×2 = 100
```

نمایش آرایه ها

حافظه کامپیوتر را می توان به صورت یک آرایه یک بعدی در نظر گرفت که آدرس ما اندیس می آن هستند. بنابراین در واقع برای نمایش آرایه در حافظه، یک آرایه n بعدی در یک آرایه یک بعدی جای داده می شود. برای این کار باید اندیس می آرایه n بعدی تبدیل به آدرس حافظه شود. دو روش برای این تبدیل وجود دارد :

- روش سطر (row-major order)
- روش ستونی (column-major order)

روش سطری

در روش سطر عناصر یک سطر پشت سر هم در حافظه قرار می گیرند در نتیجه اندیس می سمت راست سریع تر حرکت می کنند .

برای به دست آوردن آدرس هر عنصر درون حافظه کامپیوتر فقط آدرس اولین عنصر آرایه لازم است. این آدرس را آدرس شروع یا α می نامیم. با استفاده از آن آدرس محل بقیه عناصر آرایه در حافظه به دست می آید .

آرایه $a[2][3]$ را در نظر بگیرید که از آدرس α حافظه شروع شده است.

آدرس	مقدار
$\alpha + 0 \times \text{sizeof}(T)$	$a[0][0]$
$\alpha + 1 \times \text{sizeof}(T)$	$a[0][1]$
$\alpha + 2 \times \text{sizeof}(T)$	$a[0][2]$
$\alpha + 3 \times \text{sizeof}(T)$	$a[1][0]$
$\alpha + 4 \times \text{sizeof}(T)$	$a[1][1]$
$\alpha + 5 \times \text{sizeof}(T)$	$a[1][2]$

آدرس اولین عنصر سطر اول برابر با $\alpha + 0 \times \text{sizeof}(t) = \alpha$ چون در هر سطر ۳ عنصر وجود دارد بنابراین آدرس عنصر اول سطر دوم برابر با $\alpha + 3 \times \text{sizeof}(t)$ می شود .

می توان یک فرمول کلی برای آرایه n بعدی به دست آورد. اگر آرایه $A[\delta_1][\delta_2] \dots [\delta_n]$ از آدرس α حافظه شروع شده باشد. آدرس خانه $A[i_1][i_2] \dots [i_n]$ این آرایه به صورت زیر محاسبه می شود (تعداد بایت های نوع آرایه است):

$$\alpha + W [(i_1 - \delta_1) \delta_2 + (i_2 - \delta_2) \delta_3 + \dots + (i_n - \delta_n)]$$

اگر آرایه توسط زبان Pascal به صورت $A[L1..U1, L2..U2, \dots, Ln..Un]$ تعریف شده باشد فرمول به شکل زیر در می آید :

$$\alpha + W [(i1-L1)\delta_1 + (i2-L2)\delta_2 + \dots + (in-Ln)\delta_n]$$

$$\delta_i = U_i - L_i$$

مثال ۱. آدرس خانه های آرایه یک بعدی به طریق زیر محاسبه می شود.

$$A[i] = \alpha + W(i-L)$$

مثال ۲. آرایه دو بعدی A را در نظر بگیرید. اگر آرایه از آدرس ۱۰۰ ذخیره شده باشد، آدرس خانه $A[12][4]$ به طریق زیر محاسبه می شود.

A : Array [10..15][3..5] of integer;

$$\delta_1 = 15 - 10 + 1 = 6, \delta_2 = 5 - 3 + 1 = 3$$

$$A[i1, i2] = \alpha + W [(i1-L1)\delta_1 + (i2-L2)\delta_2]$$

$$A[12, 4] = 100 + 2 [(12-10)6 + (4-3)3] = 114$$

مثال ۳. اگر آرایه سه بعدی num از آدرس ۲۰۰۰ به بعد ذخیره شده باشد، آدرس خانه $A[3][2][1]$ به طریق زیر محاسبه می شود:

float A[4][5][3];

$$\delta_1 = 4, \delta_2 = 5, \delta_3 = 3$$

$$A[i1][i2][i3] = \alpha + W [(i1-L1)\delta_1 + (i2-L2)\delta_2 + (i3-L3)\delta_3]$$

$$A[3][2][1] = 2000 + 4 [(3-1)4 + (2-1)5 + (1-1)3] = 2208$$

روش ستونی

در روش ستونی عناصر یک ستون پشت سر هم در حافظه قرار می گیرند در نتیجه اندیس های ست چپسریع تر حرکت می کنند .

آدرس خانه $A[i1][i2] \dots [in]$ در این روش توسط فرمول زیر محاسبه می شود:

$$\alpha + W [(i1-L1) + \delta_1(i2-L2) + \delta_1\delta_2(i3-L3) + \dots + \delta_1\delta_2\delta_3 \dots \delta_{n-1}(in-Ln)]$$

اگر آرایه توسط زبان Pascal تعریف شده باشد فرمول به شکل زیر در می آید :

$$\alpha + W [(i1-L1) + \delta_1(i2-L2) + \delta_1\delta_2(i3-L3) + \dots + \delta_1\delta_2\delta_3 \dots \delta_{n-1}(in-Ln)]$$

$$\delta_i = U_i - L_i$$

مثال. در مثال ۲ اگر آرایه به صورت ستونی ذخیره شده باشد:

$$A[i1, i2] = \alpha + W [(i1-L1) + \delta_1(i2-L2)]$$

$$A[12, 4] = 100 + 2 [(12-10) + 6(4-3)] = 116$$

مثال. در مثال ۳ اگر آرایه به صورت ستونی ذخیره شده باشد:

$$A[i_1][i_2][i_3] = a + W[i_1] + \delta_1(i_2) + \delta_1\delta_2(i_3)$$

$$A[3][2][1] = 2000 + 4[(3) + 4(2) + 4 \times 5(1)]$$

$$= 2124$$

آرایه های پویا

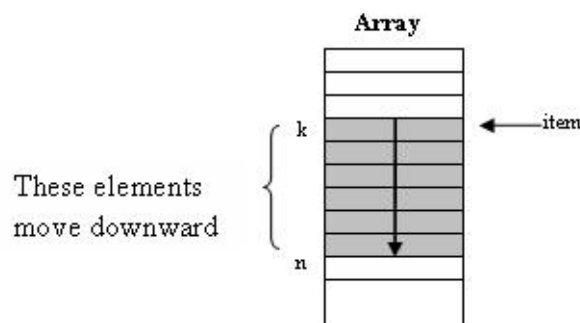
آرایه پویا (dynamic array) آرایه‌ای است که اندازه اش در زمان اجرا با عمل درج یا حذف عناصر در آن تغییر می کند.

در زبان برنامه نویسی Visual Basic توسط متور REDIM و در زبان C با تعریف حافظه پویا می توان آرایه پویا ایجاد کرد. در بعضی زبان ما مانند Perl کلیه آرایه ما به صورت پویا هستند.

الگوریتم های درج و حذف

درج عنصری در آرایه

برای درج عنصری در آرایه تعدادی از عناصر باید به سمت پائین منتقل شوند تا عنصر جدید در محل مورد نظر قرار گیرد. اگر بخواهیم عنصر جدید در مکان k ام آرایه درج شود کلیه عناصر از k به بعد باید شیفت داده شوند، سپس عنصر در مکان K ام ذخیره شود.



در کل $n-k+1$ عنصر باید جابجا شوند. اگر عنصر جدید در محل آخرین عنصر درج شود تنها عنصر آخر آرایه جابجا می شود. بدترین حالت زمانی اتفاق می افتد که بخواهیم عنصر جدید را در مکان اول آرایه درج کنیم در این حالت تعداد جابجائی ما برابر با n می شود.

به طور متوسط نیاز به $(n+1)/2$ جابجائی است.

با هر بار عمل درج یک واحد به n تعداد عناصر آرایه اضافه می شود. تعداد عناصری که در آرایه درج شده اند را نشان می دهد و ربطی به طول آرایه ندارد.

الگوریتم زیر عنصر item را در مکان k ام آرایه A با n عنصر درج می کند.

```
for (i:= n down to k )
    A[j+1] := A[j]
end for
```

```

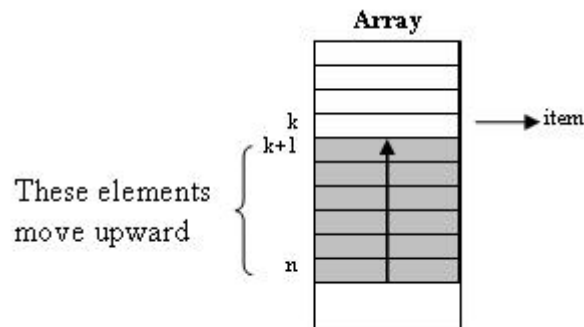
A[k] := item
n := n + 1
end

```

پیچیدگی الگوریتم فوق $O(n)$ است.

حذف عنصری از آرایه

وقتی عنصری از آرایه حذف می شود عناصر بعدی باید به سمت بالا منتقل شوند تا محل عنصر حذف شده پر شود. برای حذف عنصری که در مکان k قرار دارد، کلیه عناصر از $k+1$ به بعد باید به سمت بالا شیفت داده شوند.



در کل $n-k$ عنصر باید جابجا شوند. کمترین جابجائی وقتی است که عنصر آخر حذف شود که هیچ عنصری جابجا نمی شود. در بدترین حالت تعداد جابجائی ها برابر با $n-1$ است وقتی که اولین عنصر آرایه حذف می شود. به طور متوسط نیاز به $(n-1)/2$ جابجائی است.

با هر بار عمل درج یک واحد به n تعداد عناصر آرایه اضافه می شود n . تعداد عناصری که در آرایه درج شده اند را نشان می دهد و ربطی به طول آرایه ندارد.

الگوریتم زیر عنصر k ام آرایه A را حذف و در $item$ ذخیره می کند.

```

item := A[k]
for (j := k to n - 1)
    A[j] := A[j + 1]
end for
n := n - 1
end

```

پیچیدگی الگوریتم فوق $O(n)$ است.

همانطور که مشاهده می شود عملیات درج و حذف در آرایه به طور متوسط منجر به انتقال نصف عناصر آرایه می شود. بنابراین در مواردی که مجموعه عناصر داده ای به طور مکرر در حال اضافه و حذف هستند آرایه خطی روش آراامدی برای ذخیره داده ما نیست.