کاربرد آرایه ها در

ساختمان داده و زبان های برنامه سازی

C++ , Java , Visual Basic

نویسنده

سعید قدیری مقدم نیاری

فهرست مطالب

مقدمه

این روزها هیاهوی رایانه همه جا را فرا گرفته است. امروز تقریبا به هر جا که می روید رایانه ها خود نمایی می کنند. شاید آن روز که بیل گیتس جوان در رویاهایش یک رایانه را در هر خانه می دید، فکر نمی کرد که در کمتر از ۲۰ سال رایانه اینطور فراگیر شود. اما رایانه با این همه زرق و برق به تنهایی قادر نیست حتی پاسخ ۱+۱ را به شما بدهد. در حقیقت رایانه بدون نرم افزار یک کالبد بی روح است و کوچکترین کارایی ندارد. اینجاست که اهمیت نرم افزار ها و برنامه نویسان آشکار می گردد. البته امروزه با ارائه متد های جدید برنامه نویسی و ساختارهایی همچون .NET شاید دیگر اصطلاح برنامه نویس[۱]جای خود را به توسعه دهنده[۲]داده باشد و نیازی نباشد یک توسعه دهنده مثل سابق احتیاج به درک تمام جزئیات در تولید نرم افزار را داشته باشد و فقط با استفاده از چند ابزار یک نرم افزار پیچیده را تولید کند. اما با این وجود درک جزئیات می تواند در بکارگیری این ابزار و ایجاد ابزار های جدید و سودمند کمک شایانی به برنامه نویسان و توسعه دهندگان سیستم نماید. در این مجموعه نگاهی به آرایه ها خواهیم داشت. آرایه ها شاید در نگاه اول یک بحث ساده بنظر آید، اما ممکن نیست شما بتوانید یک نرم افزار بزرگ و پیچیده را بدون استفاده از آرایه ایجاد و مدیریت نمایید.

در این کتاب پیاده سازی آرایه ها در سه زبان برنامه نویسی C++, Java, Visual Basic انتخاب شده است. شاید بپرسید چرا؟ خوب C++ از آن جهت که یک نرم افزار قوی سیستمی بشمار می آید و در کار با سخت افزار تقریبا بی همتا است، Java به خاطر انعطاف پذیری که دارد و بر روی هر پلتفرمی قابل پیاده شدن است تا جایی که امروزه وب را تسخیر نموده و زبان Visual Basic که محبوب ترین زبان تولید کننده نرم افزار بین برنامه نویسان بوده و با وجود سادگی دارای استحکام قوی و انعطاف پذیری بالا بوده و بیش از ۷۰٪ نرم افزارهای تجاری با آن تولید شده است. در این کتاب هر بخش که لازم به کد نویسی دارد ، برای هر زبان بصورت مجزا تکرار شده است تا شما اگر قصد دارید فقط به یکی از این زبان ها مسلط شوید بتوانید فقط مطالب مربوط به آن زبان را مطالعه نمایید. مطالب ارائه شده در این کتاب و برنامه های نوشته شده تماما بر اساس تجربیات کاری که طی سالیان متمادی در طراحی نرم افزار های مختلف کسب نموده ام ارائه شده است. در نهایت از کلیه کسانی که مرا در امر نگارش این کتاب یاری کرده اند به ویژه از اساتید محترم مهندس مبینی و مهندس بکروی و همچنین از ناشر محترم کمال تشکر را دارم. همچنین از اساتید و صاحب نظران گرامی و شما خوانندگان عزیز خواهشمندم تا با ارائه پیشنهادات و انتقادات خود مرا در بهبود کمی و کیفی فعالیت های خود راهنمایی فرمایید.

سعید قدیری مقدم نیاری
۱۳۸۸
http://www.GhadiriSoft.ir

---

[۱] System Programmer

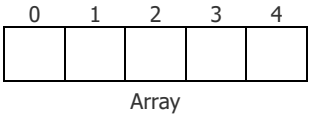
[۲] System Developer

فصل اول

- آرایه ها
- آرایه های پویا
- ارسال آرایه به توابع
- ساخت و مقدار دهی آرایه ها

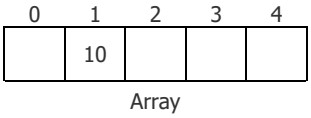
آرایه ها

یک آرایه[۳] مجموعه ای از متغیر ها[۴] ای هم نوع بوده که به صورت متوالی با یک نام از حافظه سیستم گرفته می شود. بنابراین هر آرایه دارای یک نام و چندین عنصر می باشد، که هر عنصر با یک شاخص[۵] عددی مشخص می گردد. معمولا[۶] اولین شاخص با شماره صفر شروع شده و به ترتیب افزایش می یابد.



آرایه Array دارای ۵ عنصر ، عنصر اول دارای شاخص 0، عنصر دوم دارای شاخص 1 ، ... عنصر پنجم دارای شاخص 4

هر عنصر می تواند مستقلا به کمک شاخص خود مقدار دهی شده و یا مقدار موجود در آن خوانده شود. دسترسی به این عناصر تصادفی[۷] یا مستقیم است، به این معنی که هر اندازه زمان لازم است تا اولین عنصر مقدار دهی شود، همان اندازه زمان نیاز است تا آخرین عنصر مقدار دهی شود.



برای خواندن یا نوشتن در آرایه کافیست به شاخص عنصر مورد نظر اشاره کرد برای مثال مقدار عنصر با شاخص 1 در Array معادل 10 شده است
یا به عبارتی Array(1) = 10

آدرسی که اولین عنصر آرایه به خود می گیرد را آدرس پایه[۸] می گویند و سایر عناصر متناسب با طول نوع آرایه از این آدرس فاصله می گیرند. برای مثال اگر آرایه از نوع عدد صحیح[۹] باشد، هر عنصر ۲ بایت فضا اشغال خواهند کرد،

---

[۳] Array

[۴] Variable

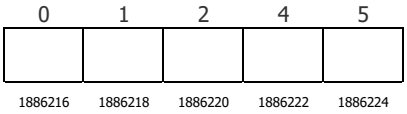
[۵] Index

[۶] در بعضی از زبان های برنامه نویسی مانند Basic می توان اولین شاخص را متناسب با نوع کاربر تغییر داد

[۷] Random

[۸] Base Address

بنابراین اگر عنصر اول دارای آدرس 1886216 باشد، عنصر بعدی در آدرس 2+1886216 یعنی آدرس 1886218 قرار خواهد گرفت.



Array : Ineger

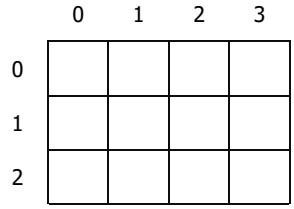
اگر B آدرس پایه و w میزان حافظه مورد نیاز هر عنصر آرایه باشد، آدرس عنصر i ام آرایه را می توان بشکل زیر محاسبه نمود:

$$Address_i = B + (w * i)$$

به عدد (w * i) که به آدرس پایه اضافه می شود تا به آدرس یک عنصر برسیم آفست[۹] آن عنصر گویند. نام یک آرایه همواره به آدرس پایه اشاره می کند، و کامپایلر[۱۱] به کمک آن مکان سایر عناصر آرایه را می تواند شناسایی کند.

آرایه های چند بعدی

آرایه های چند بعدی[۱۲] فهرستی از چند آرایه بوده، که هر بعد از آرایه با اندیس خاص مشخص می گردد. بنابراین هر عنصر یک آرایه چند بعدی متناسب با تعداد ابعاد آرایه دارای چند اندیس خواهد بود.



Array

---

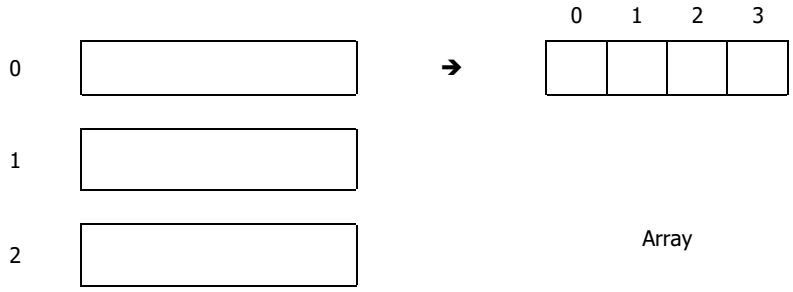
۹ Integer

۱۰ Offset

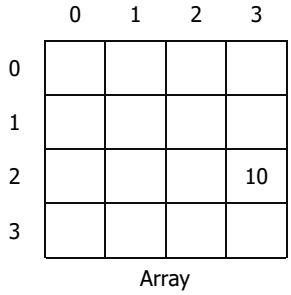
۱۱ Compiler

۱۲ Multi-Dimensional

یک آرایه دو بعدی را می توان مانند یک جدول در نظر گرفته و هر خانه جدول خود همانند یک عنصر بوده، که این عنصرها دارای ۲ شاخص ردیف و ستون می باشند. البته آرایه های بیش از یک بعد را می توان به عنوان یک آرایه که هر عنصر آن، خود یک آرایه مجزا می باشند، تجسم نمود.



همانطور که در شکل نشان داده شده است، آرایه دو بعدی Array را می توان ابتدا یک آرایه یک بعدی با ۳ عنصر در نظر گرفت که هر عنصر خود دارای ۴ عنصر داخلی هستند.
برای مقدار دهی و یا خواندن اطلاعات داخل هر عنصر به هر دو شاخص نیاز خواهیم داشت تا مکان عنصر را در آرایه مشخص نماییم.



عنصر موجود در سطر 2 و ستون 3 در Array معادل با 10 قرار گرفته است، یا به عبارتی Array (2,3) = 10

اگر B آدرس پایه آرایه باشد و L فضای اشغال شده توسط هر سطر ، به کمک فرمول زیر می توان آدرس پایه هر سطر b(j) را محاسبه کرد.

$$Address_{b(j)} = B + (L * j)$$

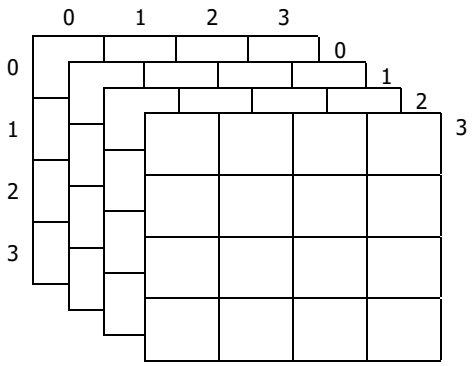
فضای اشغال شده در هر سطر نیز باید بصورت جداگانه محاسبه شود. اگر هر سطر را یک آرایه یک بعدی با n عنصر فرض کنیم، و فضای اشغال شده هر عنصر آن برابر با w باشد، در آنصورت خواهیم داشت :

$$L = w * n$$

بنابراین برای پیدا کردن آدرس یک سلول در آرایه دو بعدی می توان به این شکل عمل نمود.

$$Address_{(i,j)} = b(j) + (w*i)$$

آرایه سه بعدی مانند چندین آرایه دو بعدی روی هم قرار گرفته است، که هر آرایه دو بعدی خود با یک شاخص در آرایه سه بعدی مشخص می گردد. در اینجا هر عنصر با ۳ شاخص قابل دسترس خواهد بود. برای مثال Array(2,3,1) به یک عنصر در سطر ۲ ، ستون ۳ در صفحه ۱ اشاره می کند.



برای محاسبه آدرس هر سلول در آرایه سه بعدی مانند آرایه های دو بعدی عمل می کنیم. یعنی از بیرونی ترین شاخص شروع به محاسبه نموده تا به داخلی ترین شاخص برسیم. یعنی باید ابتدا آدرس پایه هر صفحه را محاسبه کنیم، که خود مستلزم محاسبه فضای اشغالی توسط هر صفحه P و آن هم مستلزم محاسبه فضای اشغالی توسط هر سطر L می باشد.

$$Address_{b(k)} = B + (P * K)$$

$$P = L * nL$$

$$L = w * n$$

در این فرمول K شماره صفحه و nL تعداد سطر های موجود در یک صفحه می باشد. در نهایت آدرس هر سلول به کمک فرمول زیر محاسبه می گردد.

$$Address_{(i,j,k)} = b(k) + (L*j) + (w*i)$$

برای محاسبه آرایه هایی با ابعاد بیشتر، دقیقا با همین روش می توان آدرس هر سلول را محاسبه نمود، که با افزایش هر بعد یک مرحله عملیات افزایش می یابد. تعداد ابعاد یک آرایه از لحاظ نرم افزاری محدودیتی ندارد، و تنها ممکن است محدود بودن حافظه فیزیکی سیستم مانع از ایجاد آرایه هایی با ابعاد بسیار بزرگ شود، چرا که هر قدر ابعاد یک آرایه افزایش می یابد، مقدار فضای بیشتری برای آن نیاز خواهد بود.

ساخت و مقدار دهی آرایه در C++

ایجاد یک آرایه همانند ایجاد یک متغیر است، با این تفاوت که تعداد عناصر آن با کمک عملگر [] مشخص می گردد.

```
Element_type array_name [index];
```

Element_type مشخص کننده نوع متغیر است و اگر بخواهیم یک آرایه چند بعدی ایجاد کنیم، به تعداد هر بعد از یک عملگر [] استفاده خواهیم کرد. array_name نام آرایه بوده که دقیقا از قانون نام گذاری متغیرها تبعیت میکند.

```
Element_type array_name [index1][index2][index3];
```

Index داخل عملگر [] تعداد عنصر های موجود در هر بُعد آرایه را مشخص می کند. در این حالت شماره اولین شاخص بصورت پیشفرض صفر خواهد بود، بنابراین اگر تعداد یک شاخص را معادل با 3 قرار دهیم آرایه دارای ۳ عنصر 0 و 1 و 2 خواهد بود.
دقت کنید تعداد عناصر یک آرایه برابر است با عددی که به عنوان index تعریف می کنیم. اگر index برابر n باشد، آرایه دارای n عنصر خواهد بود اما شاخص آخرین عنصر برابر با n-1 خواهد بود. یعنی اگر int a[3] داشته باشیم، یک آرایه با 3 عنصر ایجاد می کند ولی شاخص آخرین عنصر معادل3-1 یعنی ۲ خواهد بود.

int a [**3**] ;	➔	a[0]	a[1]	a[2]	
int b [**2**][**2**];	➔	b[0][0]	b[1][0]	b[0][1]	b[1][1]

زبان C++ این امکان را می دهد تا همزمان با ایجاد یک آرایه عناصر آن را نیز مقدار دهی نماییم. برای این منظور مقادیر مورد نظر را در داخل **{}** قرار داده و هر مقدار را با , از یک دیگر مجزا میکنیم.

```
Element_type array_name [index]={value1,value2,…};
```

اگر تعداد مقادیر اعلان شده کمتر از تعداد عناصر مشخص شده توسط index باشد، سایر عناصر مقدار خالی دریافت خواهند کرد. اگر نوع متغیر عددی باشد مقدار 0 و اگر کاراکتری باشد[۱۳]\0 در عناصر بدون مقدار قرار خواهد گرفت.
حال اگر بخواهیم یک آرایه با عناصر خالی[۱۴]ایجاد کنیم، آرایه را در هنگام ایجاد بشکل زیر مقدار دهی می کنیم.

```
Element_type array_name={0};
Element_type array_name={'\0'};
```

مثال زیر آرایه ای ۱۰۰ عنصری ایجاد می کند که تمام عناصر آن معادل با صفر است.

```
int b[100]={0};
```

در طول برنامه نیز با مشخص کردن شاخص آرایه می توان عناصر متناظر را مقدار دهی نمود.

```
int a [3]={10,20,30};
double b [6][2];                    //آرایه دو بعدی از نوع اعشاری با ۱۲ عنصر (۲ سطر و هر سطر ۶ سلول)
b [3][0] = 3.1415;                  //خانه ای که در سطر ۱ و ستون ۴ قرار دارد با مقدار ۳,۱۴۱۵ مقدار دهی می شود.
```

---

[۱۳] کاراکتر تهی که کد آسکی آن برابر با صفر است

[۱۴] زبان C هنگامی که بلوکی از حافظه را برای یک متغیر در نظر می گیرد مقدار داخل آن را خالی نمی کند، بنابراین ممکن است متغیر ایجاد شده داری مقداری باشد که به آن زباله گفته می شود

```
For (int i =0 ; i<3 ; i++ ) {
        b [i][1]=10.1;          //در طول این حلقه تمام عناصر موجود در سطر دوم را با عدد ۱۰,۱ مقدار دهی می کند
        cout << b[i][1] ; }     //سپس مقادیر موجود در آنها را چاپ می کند
```

اگر آرایه از نوع کاراکتری[۱۵]باشد مقدار دهی عناصر آن مقداری متفاوت خواهد بود. می دانیم که در زبان C++ به کمک علامت ' ' می توان یک کاراکتر و علامت " " یک رشته را مشخص نمود. پس در مقدار دهی آرایه نیز باید این موضوع را مد نظر داشت. و هر کاراکتر را مابین علامت ' درج کنیم.

```
char n[10]={'s','a','e','e','d'};
```

➔ n[0]='s' n[1]='a' n[2]='e' n[3]='e' n[4]='d'

در دستور بالا عنصر ها با شاخص صفر تا ۴، کاراکتر های مشخص شده را در خود ذخیره کرده و سایر عنصر ها با کاراکتر تهی مقدار دهی می گردند. اما نکته جالب اینجاست که اگر در هنگام مقدار دهی اولیه یک رشته را به آرایه نسبت دهیم، رشته به کاراکترهای تشکیل دهنده شکسته شده و هر کاراکتر وارد یک عنصر می شود. به مثال زیر دقت کنید:

```
char n[10]={"saeed"};
```

➔ n[0]='s' n[1]='a' n[2]='e' n[3]='e' n[4]='d'

در این حالت نیز دقیقا عنصر ها با شاخص صفر تا ۴ با تک تک کاراکتر های رشته مورد نظر مقدار دهی شده اند. باید دقت کنید تعداد کاراکتر های رشته وارد شده نباید بیشتر از تعداد عناصر تعریف شده در آرایه باشد، این موضوع باعث می شود تا کامپایلر با خطا متوقف گردد.
تعداد عناصر موجود در یک آرایه را می توان با محاسبه فضای اشغال شده توسط آرایه، و تقسیم آن بر مقدار فضایی که برای هر عنصر نیاز است بدست آورد. برای این منظور از عملگر sizeof می توان استفاده نمود.

```
index_count = sizeof (array_name) / sizeof (element_type);
```

مثال :

```
float a[2]={3.14,1.97};
int b=0;
b = sizeof (a) / sizeof (float);
cout << b ;          ➔          2
```

می دانیم هر متغیر float مقدار ۴ بایت فضا اشغال می کند بنابراین sizeof(float) مقدار ۴ را بر می گرداند. آرایه a نیز از نوع float بوده و ۲ عنصر دارد پس مقدار ۸ بایت فضا را اشغال کرده است. پس sizeof(a) نیز مقدار ۸ را بر می گرداند. که حاصل تقسیم ۸ بر ۴ عدد ۲ و همان تعداد عناصر آرایه a است.

---

[۱۵] Character

ساخت ، مقدار دهی آرایه در Visual Basic

یک آرایه در بیسیک همانند یک متغیر به کمک دستور Dim ایجاد می گردد، تنها تفاوت آن در استفاده از عملگر () است، که تعداد عناصر آرایه را اعلان می کند.

Dim array_name (index) **As** element_type

array_name نام آرایه بوده که دقیقا از همان قوانین نامگذاری متغیر ها تبعیت می کند و element_type نوع آرایه را مشخص می کند. همانند متغیر برای اعلان نوع آرایه از عبارت As استفاده می کنیم. اگر قصد ایجاد یک آرایه چند بعدی را داشته باشیم، می توان هر بُعد را در داخل عملگر ( ) و با کمک , از یکدیگر متمایز کرد.

Dim array_name (index1,index2,index3) As element_type

شاخص بندی عناصر آرایه در بیسیک متفاوت با اکثر زبان های برنامه نویسی می باشد. عددی را که در index مشخص می کنیم در حقیقت شاخص آخرین عنصر قلمداد می گردد. و اولین عنصر شاخص صفر را می گیرد. بنابراین اگر index را معادل با 3 قرار دهیم آرایه دارای 4 عنصر 0 و 1 و 2 و 3 خواهد بود، اگر index را برابر با n بگیریم تعداد عناصر آرایه برابر n+1 خواهد شد. یعنی Dim a(3) یک آرایه با 3+1 و به عبارتی 4 عنصر ایجاد می کند.

Dim a (3) as integer	➔	a(0)	a(1)	a(2)	a(3)
Dim b(2,2) as integer	➔	b(0,0)	b(1,0)	b(2,0)	
		b(0,1)	b(1,1)	b(2,1)	
		b(0,2)	b(1,2)	b(2,2)	

شاید کمی گیج کننده بنظر بیاید، به همین خاطر بیسیک به شما این اجازه را می دهد تا پیشفرض شماره دهی به عناصر آرایه را از صفر به یک تغییر دهید. کافیست از دستور Option Base در بخش General_Declarations استفاده نمایید.[۱۶]

Option Base 1/0

اگر مقدار 1 را انتخاب کنید، پیشفرض اولین شاخص در آرایه ها شماره ۱ خواهد بود، و اگر مقدار 0 را انتخاب کنید شماره اولین شاخص معادل با صفر خواهد شد. باید به این نکته توجه کنید که این تعریف بر روی تمام آرایه ها اعمال می گردد.
اگر مقدار 1 را انتخاب نمایید، تعداد عناصر یک آرایه برابر n خواهد بود، یعنی Dim a(3) یک آرایه با ۳ عنصر ایجاد می کند که شاخص اولین عنصر آن برابر با ۱ خواهد بود.

Option Base 1

Dim a (3) as integer	➔	a(1)	a(2)	a(3)

[۱۶] اگر این دستور را در داخل توابع یا روال ها استفاده کنید، کامپایلر با اشکال مواجه خواهد شد.

در بیسیک تا نسخه Visual Basic 6 نمی توان همزمان با تعریف آرایه آن را مقدار دهی کرد. به همین دلیل بیسیک بصورت خود کار فضای درون بلوک های حافظه را که برای آرایه در نظر گرفته است خالی می کند تا در طول برنامه با داده های زباله مواجه نشویم. مقدار دهی آرایه در طول برنامه امکان پذیر است.

مثال :

```
Dim a (2) as Integer                    'آرایه یک بعدی از نوع عدد صحیح با ۳ عنصر
Dim b (5,1) as Double                   'آرایه دو بعدی از نوع اعشاری با ۱۲ عنصر (۲ سطر و هر سطر ۶ سلول)
a(0) = 10                               'عنصر با شاخص صفر در داخل آرایه دارای مقدار ۱۰ می شود
a(2) = 30                               'عنصر با شاخص ۲ در داخل آرایه دارای مقدار ۳۰ می شود
for i = 0 to 5
        b(i,1)=10.1                     'در طول حلقه تمام ۶ عنصر در سطر دوم با مقدار ۱۰.۱ مقدار دهی می شود
        debug.print b(i,1)              ' سپس مقادیر داخل آرایه چاپ می شود
next
```

یکی از قابلیت های منحصر به فرد بیسیک در مدیریت شاخص های عناصر آرایه می باشد. همانطور که در بالا دیدید وقتی برای یک آرایه شاخص را معادل n قرار می دهیم، بصورت پیشفرض حد پایین معادل صفر و حد بالا معادل n می گردد.

```
Dim array_name (n) As element_type  ➔        L = 0    H = n
```

حال ممکن است بخواهیم حد پایین L و حد بالا H را خود تعریف کنیم، برای این منظور می توان آرایه را بصورت زیر تعریف کنیم.

```
Dim array_name ( L To H ) As element_type
```

در این حالت تعداد عناصر تعریف شده (H – L) + 1 خواهد بود.

مثال :

```
Dim a ( 5 to 8 ) as ineger  ➔    L = 5    H = 8    n = 4
```

در مثال بالا آرایه a از نوع عددی ایجاد می گردد در حالی که شاخص اولین عنصر آن ۵ و شاخص آخرین عنصر آن ۸ خواهد بود. در مجموع این آرایه دارای ۴ عنصر به قرار زیر است.

```
➔    a(5)    a(6)    a(7)    a(8)
```

در طول برنامه می توان به کمک دستور UBound شاخص آخرین عنصر (حد بالا) و دستور LBound شاخص اولین عنصر (حد پایین) آرایه مورد نظر را بدست آورد.

```
UBound ( array_name , index_dimensional )
LBound ( array_name , index_dimensional )
```

index_dimensional ، به بُعد آرایه اشاره می کند، در آرایه یک بُعدی می توان از آن چشم پوشی نمود، اما اگر آرایه چند بُعدی باشد، به بُعد آرایه شاره می کند.

```
Dim a ( 5 to 8 ) as ineger
        debug.print LBound ( a )    ➔    5
        debug.print UBound ( a )    ➔    8

Dim a ( 1 to 3 , 2 to 5) as integer
        debug.print LBound ( a , 1)    ➔    1
        debug.print UBound ( a , 1)    ➔    3

        debug.print LBound ( a , 2)    ➔    2
        debug.print UBound ( a , 2)    ➔    5
```

دستور Erase برای پاک کردن مقادیر موجود در آرایه مورد استفاده قرار می گیرد. این دستور فضای اختصاص داده شده به آرایه را آزاد نمی کند، بلکه مقادیر موجود در این حافظه ها را خالی می کند.

```
Erase array_name
```

مثال :

```
Dim a (1) as integer
        a (0) = 100
        a (1) = 200
        debug.print a (0) , a (1)    ➔    100    200
        Erase a
        debug.print a (0) , a (1)    ➔    0    0
```

آرایه های پویا[۱۷]

در برنامه هایی که می نویسیم ممکن است با مواردی روبرو شویم که تعداد عناصر آرایه نامشخص باشد، و دقیقا ندانیم آرایه ای که ایجاد می کنیم به چه مقدار عنصر نیاز دارد، و یا اینکه در طول برنامه ممکن است لازم باشد تعداد عناصر آرایه را افزایش یا کاهش دهیم. در حقیقت ممکن است در طول برنامه نیاز داشته باشیم تا فضای اختصاص داده شده به آرایه را آزاد کنیم و یا حافظه بیشتری را در اختیار آرایه قرار دهیم. نوع تعریف آرایه که در صفحات قبل اشاره شد، آرایه ها را بصورت استاتیک تعریف می کرد، و امکان تغییر در تعداد عناصر را نمی داد. برای این منظور به آرایه های پویا نیاز خواهیم داشت.

---

[۱۷] Dynamic