فصل 12

آرایه ها

* آرایه ها
* انواع آرایه ها
* آرایه ای به عنوان شی
* آرایه های یک بعدی و مستطیلی
* مقدار اولیه یک آرایه یک بعدی یا مستطیلی
* دسترسی به عناصر آرایه
* مقداردهی اولیه یک آرایه
* آرایه های دندانه دار
* مقایسه آرایه های مستطیلی و دندانه دار
* دستور foreach
* کواریانس آرایه
* عضو هایی مفید از ارث بری آرایه ها
* مقایسه انواع آرایه ها

آرایه ها

آرایه مجموعه از عناصر داده های یکنواخت ارائه شده توسط نام یک متغییر است. عناصر منحصر به فرد با استفاده از نام متغییر به همراه یک یا چند شاخص بین کروشه قابل دسترس هستند. همانطور که در زیر نشان داده شده است:

نام آرایه شاخص

↓ ↓

MyArray[4]

تعاریف

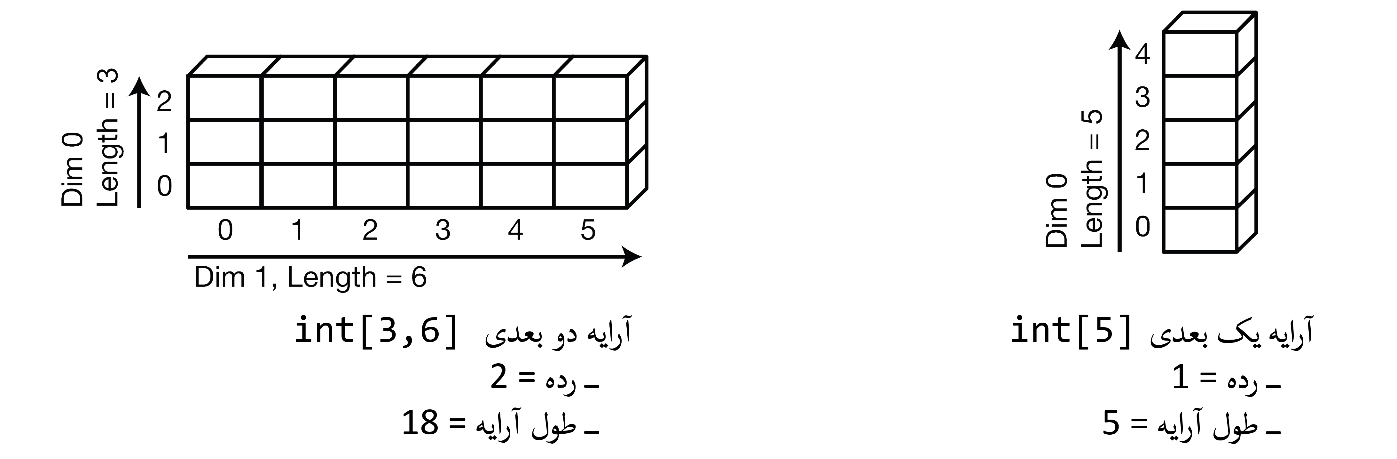
بیایید با برخی از تعاریف مهم برای کار با آرایه ها در C# شروع کنیم.

* عناصر[[1]](#footnote-1) : آیتم های داده منحصر به فردی از یک آرایه را عناصر نامیده می شوند. همه عناصر یک آرایه باید از همان نوع و یا مشتق شده از همان نوع باشد.
* رده بندی[[2]](#footnote-2) / بُعد[[3]](#footnote-3) : آرایه های می توانند هر عدد مثبتی از ابعاد را داشته باشد. تعداد ابعاد یک آرایه رده آن نامیده می شود.
* طول بعد[[4]](#footnote-4) : هر بعد یک آرایه طول دارد، که تعداد موقعیت های آن جهت است.
* طول آرایه[[5]](#footnote-5) : تعداد کل عناصر موجود در یک آرایه در کل ابعاد، طول آرایه نامیده می شود.

جزئیات مهم

در زیر برخی از موارد مهم در مورد آرایه های زبان C# وجود دارد:

* هنگامی که یک آرایه ایجاد می شود، اندازه آن ثابت خواهد بود. C# آرایه های پویا را پشتیبانی نمی کند.
* شاخص آرایه ها از صفر هستند. به این معنا که، اگر طول یک بعد n باشد، محدوده شاخص از 0 تا n-1 خواهد بود. به عنوان مثال، شکل 1-12 ابعاد و طول دو آرایه نمونه را نشان می دهد. توجه کنید که برای هر یک از ابعاد محدوده شاخص بین 0 تا length – 1 است.



شکل 1-12 ابعاد و اندازه

انواع آرایه ها

C# دو نوع آرایه ها را ارائه می دهد:

* آرایه یک بعدی می توان همانند یک خط یا برداری از عناصر تصور کرد.
* آرایه های چند بعدی تشکیل شده اند که هر موقعیت در بردار اولیه خود یک آرایه است که زیر آرایه نامیده می شود. موقعیت بردارهای زیر آرایه می توانند خود زیر آرایه باشند.

به علاوه، آرایه های مستطیلی و آرایه های دندانه دار که آرایه هایی چند بعدی دارای ویژگی های زیر هستند:

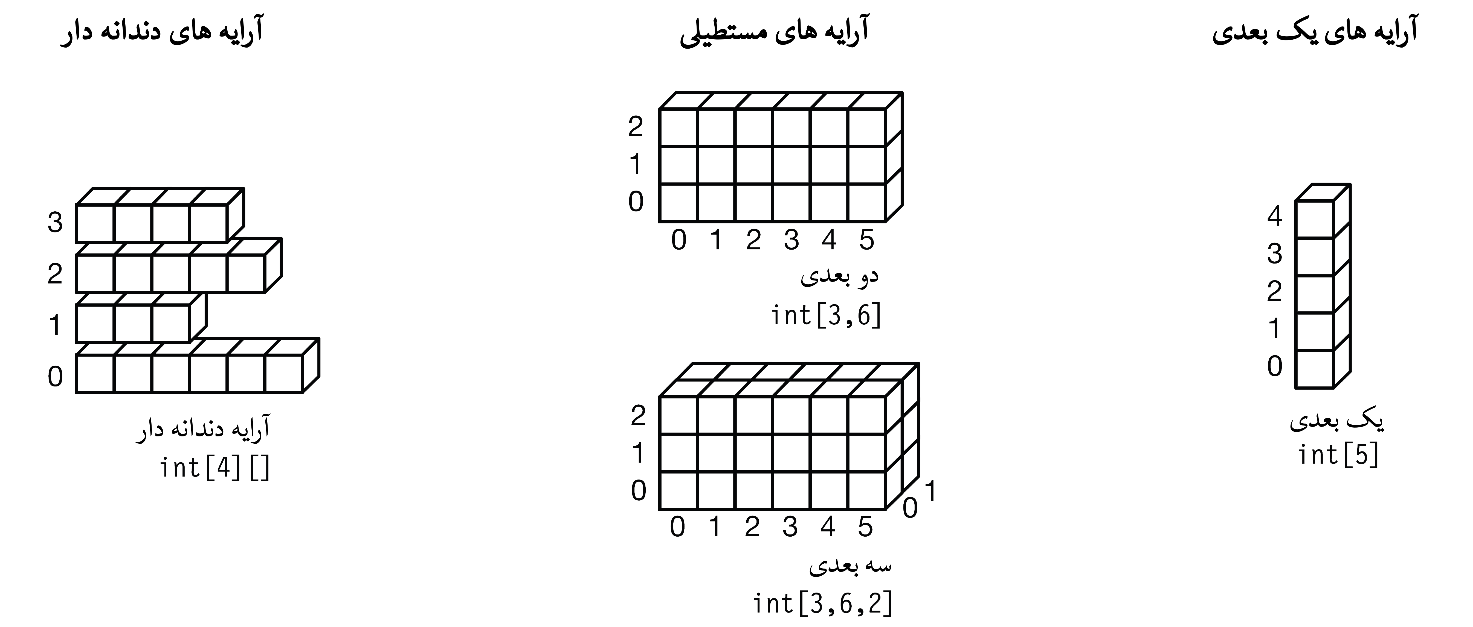
* آرایه های مستطیلی[[6]](#footnote-6)
  + آرایه هایی چند بعدی هستند که تمام زیر آرایه ها در یک بعد خاص دارای همان طول هستند.
  + همیشه از یک کروشه بدون در نظر گرفتن تعداد ابعاد استفاده می کنند.

int x = myArray2[4, 6, 1] // یک کروشه

* آرایه های دندانه دار[[7]](#footnote-7)
  + آرایه هایی چند بعدی هستند که هر زیر آرایه، آرایه ای مستقل است.
  + می توانند زیر آرایه هایی با طول های متفاوت داشته باشند.
  + از یک براکت جداگانه برای هر یک از ابعاد آرایه استفاده می کنند.

jagArray1[2][7][4] // سه کروشه

شکل 2-12 انواع آرایه های را در زبان C# نشان می دهد.



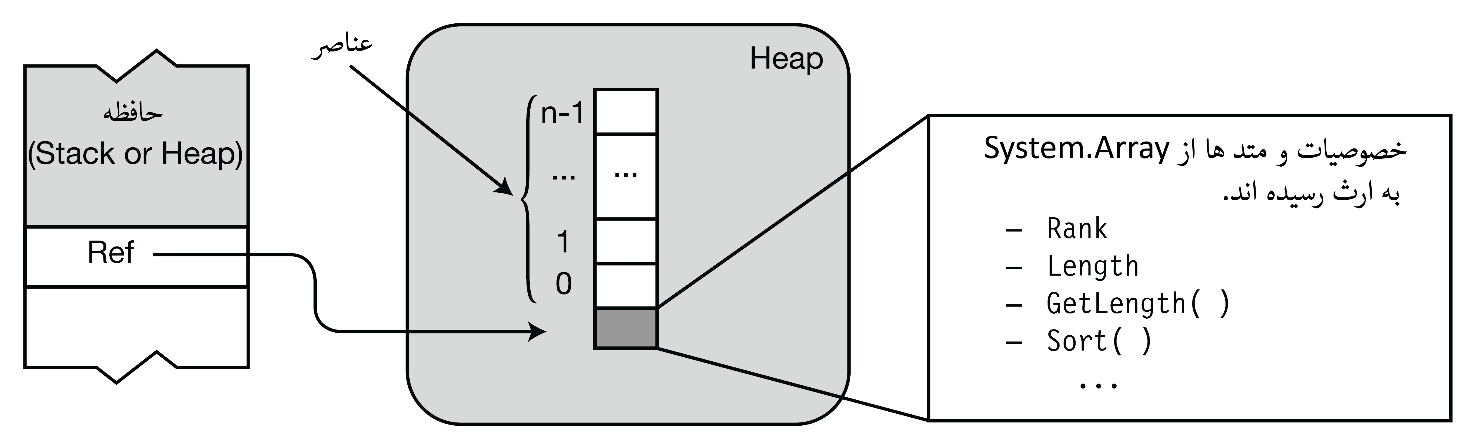
شکل 2-12 آرایه های یک بعدی، مستطیلی شکل و دندانه دار

آرایه ای به عنوان شی

یک آرایه نمونه سازی شده شی ای است که از کلاس System.Array مشتق شده است. از آنجایی که آرایه ها از کلاس پایه BCL مشتق شده اند، آنها تعدادی از عضو های مفید را از آن به ارث می برند. نظیر موارد زیر:

* Rank: خصوصیتی که تعداد ابعاد آرایه را برمی گرداند.
* Length: خصوصیتی که طول (تعداد کل عناصر) آرایه را برمی گرداند.

آرایه ها نوع های ارجاعی هستند، و همانند تمام نوع های ارجاعی، آنها هم مرجع برای داده ها و هم مرجعی برای داده های شی خود دارند. مرجع یا در stack یا در heap است، و داده های شی آن همیشه در heap است. شکل 3-12 پیکر بندی حافظه و اجزای آرایه را نشان می دهد.

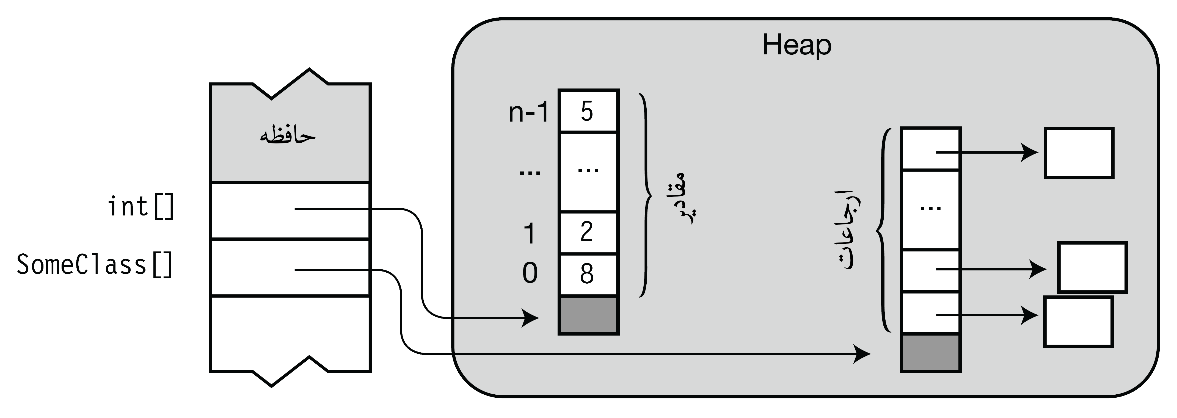


شکل 3-12 ساختار یک آرایه

اگر چه یک آرایه همیشه یک نوع ارجاعی است، اما عناصر آرایه می توانند هم از نوع مقداری و هم از نوع ارجاعی باشند.

* یک آرایه از نوع مقداری است اگر عناصر ذخیره شده در آن از نوع مقداری باشد.
* یک آرایه از نوع ارجاعی است اگر عناصر ذخیره شده در آرایه مرجع یک شی ارجاعی باشد.

شکل 4-12 آرایه ای از نوع مقداری و آرایه ای از نوع ارجاعی را نشان می دهد.



شکل 4-12 عناصر می توانند از نوع مقداری یا از نوع ارجاعی باشند.

آرایه های یک بعدی و مستطیلی

آرایه های یک بعدی و آرایه های مستطیلی بسیار شبیه به هم هستند، بنابراین آنها را با یکدیگر مورد بحث قرار می دهیم. درباره آرایه های دندانه دار را به طور جداگانه بحث خواهیم کرد.

تعریف یک آرایه یک بعدی یا مستطیلی

برای تعریف یک آرایه یک بعدی یا مستطیلی شکل، از یک جفت کروشه بین نوع و نام متغییر استفاده می شود.

تعیین کننده های رده[[8]](#footnote-8) ویرگول ها هستند که بین کروشه قرار می گیرند. آنها تعداد ابعاد آرایه را مشخص می کنند. تعداد رده ها برابر تعداد ویرگول ها بعلاوه یک است. به عنوان مثال، نبودن ویرگول نشان دهنده آرایه یک بعدی، یک ویرگول نشان دهنده نشان دهنده آرایه دو بعدی و ... .

نوع پایه، همراه با تعیین کننده های رده، نوعی آرایه هستند. به عنوان مثال، کد زیر آرایه ای یک بعدی از نوع Long را تعریف می کند. نوع آرایه Long[] است که به عنوان "آرایه ای از Long ها" خوانده می شود.

تعیین کننده های رده = 1

↓

long[ ] secondArray;

↑

نوع آرایه

کد زیر نمونه هایی از تعریف آرایه های مستطیلی را نشان می دهد. به موارد زیر توجه کنید:

* می توان هر تعداد تعیین کننده رده که نیاز دارید، داشته باشید.
* نمی توان طول آرایه را در بخش نوع آرایه تعیین کرد. رده بخشی از نوع آرایه است، اما طول ابعاد بخشی از این نوع نیست.
* وقتی یک آرایه تعریف شده است، تعداد ابعاد ثابت است. با این حال تا زمانی که آرایه نمونه سازی نشود، طول ابعاد مشخص نیست.

تعیین کننده های رده

↓

int[,,] firstArray; // int نوع آرایه: آرایه سه بعدی از نوع

int[,] arr1; // int نوع آرایه: آرایه دو بعدی از نوع

long[,,] arr3; // long نوع آرایه: آرایه سه بعدی از نوع

↑

نوع آرایه

long[3,2,6] SecondArray; // اشتباه! خطای کامپایل

↑ ↑ ↑

اجازه تعیین طول ابعاد وجود ندارد

|  |
| --- |
| * توجه برخلافC/C++ در C# براکت بعد از نوع پایه می آید نه بعد از نام متغییر. |

مقدار اولیه یک آرایه یک بعدی یا مستطیلی

برای نمونه سازی یک آرایه، از دستور ایجاد آرایه استفاده می شود. دستور ایجاد آرایه شامل عملگر new به همراه نوع پایه و به دنبال آن یک جفت کروشه است. طول هر یک از ابعاد در داخل کروشه با کاما از یک دیگر جدا می شوند.

نمونه هایی از تعریف آرایه یک بعدی به شرح زیر است.

* آرایه arr2 یک آرایه یک بعدی با چهار نوع داده int است.
* آرایه mcArr یک آرایه یک بعدی با چهار نوع ارجاعی از MyClass است.

شکل 5-12 نحو قرار گیری آنها را در حافظه نشان می دهد.

چهار آیتم

↓

int[] arr2 = new int[4];

MyClass[] mcArr = new MyClass[4];

↑

دستور ایجاد آرایه

در زیر مثالی از آرایه مستطیلی وجود دارد.

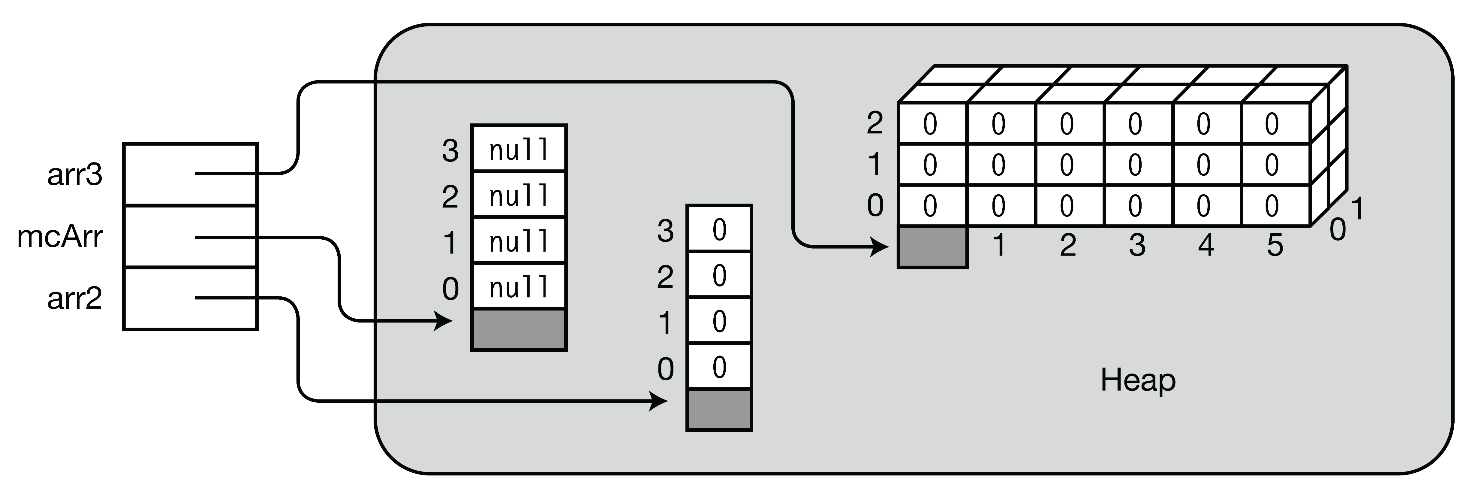
* آرایه arr3 یک آرایه سه بعدی است.
* طول آرایه برابر است.

شکل 5-12 نحو قرار گرفتن آن را در حافظه نشان می دهد.

طول ابعاد

↓

int[,,] arr3 = new int[3,6,2] ;



شکل 5-12 تعریف و مقدار دهی آرایه ها

|  |
| --- |
| * توجه برخلاف دستور ایجاد شی، دستور ایجاد آرایه حتی برای نوع آرایه های ارجاعی هم شامل پرانتز نمی باشد. |

دسترسی به عناصر آرایه

عناصر یک آرایه با استفاده از یک عدد صحیح به عنوان شاخص آرایه قابل دسترس هستند.

* هر بعد از شاخص 0 شروع می شود.
* این شاخص بعد از نام آرایه بین کروشه قرار گرفته می شود.

کد زیر نمونه هایی از تعریف، نوشتن و خواندن از آرایه های یک بعدی و دو بعدی را نشان می دهد.

int[] intArr1 = new int[15]; // تعریف آرایه یک بعد با 15 عنصر

intArr1[2] = 10; // نوشتن بر روی عنصر شماره 2 آرایه

int var1 = intArr1[2]; // خواندن از روی عنصر شماره 2 آرایه

int[,] intArr2 = new int[5,10]; // تعریف آرایه دو بعدی

intArr2[2,3] = 7; // نوشتن بر روی آرایه

int var2 = intArr2[2,3]; // خواندن از روی آرایه

کد زیر روندی کامل از ایجاد و دسترسی به یک آرایه یک بعدی را نشان می دهد:

int[] myIntArray; // تعریف آرایه

myIntArray = new int[4]; // نمونه سازی آرایه

for( int i=0; i<4; i++ ) // تعیین مقدار

myIntArray[i] = i\*10;

// خواندن و نمایش مقادیر هر یک از عناصر

for( int i=0; i<4; i++ )

Console.WriteLine("Value of element {0} = {1}", i, myIntArray[i]);

این کد خروجی زیر را تولید می کند:

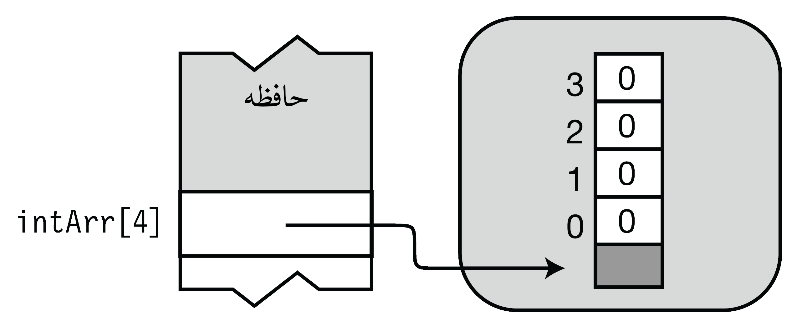
|  |
| --- |
| Value of element 0 is 0  Value of element 1 is 10  Value of element 2 is 20  Value of element 3 is 30 |

مقدار دهی اولیه یک آرایه

هر گاه یک آرایه ایجاد می شود، هر یک از عناصر به طور خودکار با مقدار پیش فرض آن نوع مقدار دهی اولیه می شود. مقادیر پیش فرض برای نوع های صحیح 0 و برای نوع های اعشاری 0.0 و برای نوع های بولین false و برای نوع های ارجاعی null هستند.

به عنوان مثال، کد زیر یک آرایه ایجاد می کند و با چهار عنصر با مقدار 0 مقدار دهی شده است. شکل 6-12 نحو قرار گرفتن آن را در حافظه نشان می دهد.

int[] intArr = new int[4];



شکل 6-12 مقدار دهی اولیه خودکار عناصر یک آرایه یک بعدی

مقداردهی صریح آرایه های یک بعدی

برای یک آرایه یک بعدی، می توان مقادیر اولیه را به صراحت بلافاصله بعد از دستور ایجاد آرایه تعیین کرد.

* مقدار دهی اولیه با یک توسط ویرگول از یک دیگر جدا شوند و در داخل آکولاد قرار گرفته باشد.
* طول ابعاد اختیاری هستند، چرا که کامپایل می تواند طول را از تعداد اولیه مقادیر استنباط کند.
* توجه داشته باشید که هیچ چیزی دستور ایجاد آرایه و مقداردهی اولیه لیست را از هم جدا نمی کند. به این معنا که هیچ علامت مساوی و یا عملگر اتصالی وجود ندارد.

به عنوان مثال، کد زیر آرایه های را ایجاد می کند و چهار عنصر آن را با مقادیر داخل آکولاد مقداردهی اولیه می کند. شکل 7-12 نحو قرار گیری آن را در حافظه نشان می دهد.

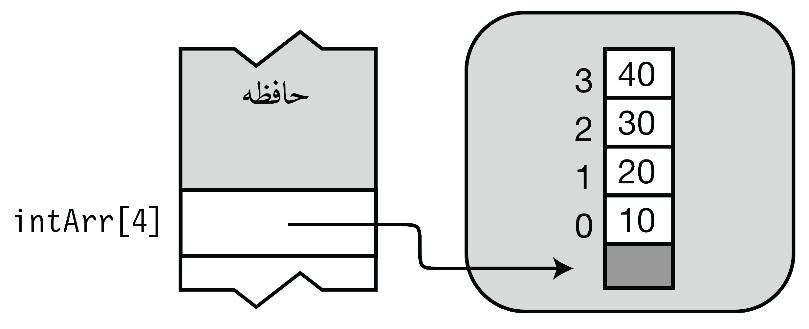
مقداردهی اولیه لیست

↓

int[] intArr = new int[] { 10, 20, 30, 40 };

↑

بدون عملگر اتصال



شکل 7-12 مقداردهی صریح آرایه یک بعدی

مقداردهی صریح آرایه های مستطیلی

برای مقداردهی صریح یک آرایه مستطیلی، نیاز است که از موارد زیر پیروی کنید:

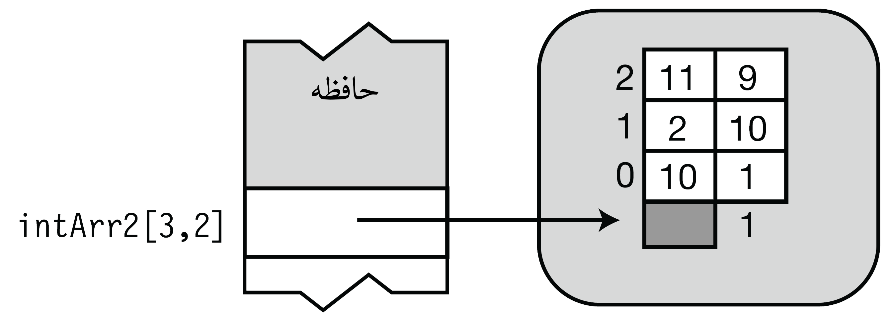
* هر مسیر از مقدار اولیه باید در آکولاد قرار گیرد.
* هر بعد باید تو در تو در داخل آکولاد قرار گیرد.
* به علاوه مقادیر اولیه، لیست های مقداردهی اولیه و اجزای هر بعد باید با کاما از یک دیگر جدا شوند.

به عنوان مثال، کد زیر یک آرایه دو بعدی با لیست مقداردهی اولیه را نشان می دهد. شکل 8-12 نحو قرار گرفتن آن را در حافظه نشان می دهد.

لیست های مقداردهی اولیه توسط کاما از هم جدا شده اند

↓ ↓

int[,] intArray2 = new int[,] { {10, 1}, {2, 10}, {11, 9} } ;



شکل 8-12 مقداردهی اولیه آرایه مستطیلی

دستورالعمل اشاره برای مقداردهی اولیه آرایه های مستطیلی

آرایه های مستطیلی با لیست های مقداردهی اولیه تودرتو مقداردهی اولیه می شوند. لیست های مقداردهی اولیه در آکولاد های تودرتو هستند. گاهی اوقات این می تواند گیج کننده باشد، بنابراین برای تودرتو کردن ، گروه بندی کردن و قرار دادن کاما صحیح، به موارد زیر توجه کنید:

* کاما ها به عنوان جدا کننده ای در بین تمام عناصر و گروه ها استفاده می شود.
* کاما ها هرگز بین آکولاد سمت چپ قرار نمی گیرد.
* کاما ها هرگز قبل از آکولاد سمت راست قرار نمی گیرد.
* در صورت امکان، با استفاده از دندانه دار کردن گروه ها، آنها را به طور بصری از یک دیگر متمایز کنید.
* مشخصات رده را از چپ به راست بخوانید، شماره قبلی را به عنوان "عناصر" و تمام آنها را با هم به عنوان "گروه" تعیین کنید.

به عنوان مثال، تعریف زیر به عنوان " intArray دارد چهار گروه از سه گروه از دو عنصر" خوانده می شود.

لیست های مقداردهی، تودرتو و با کاما از یک دیگر جدا شده اند

int[,,] intArray = new int[4,3,2] { ↓ ↓ ↓

{ {8, 6}, {5, 2}, {12, 9} },

{ {6, 4}, {13, 9}, {18, 4} },

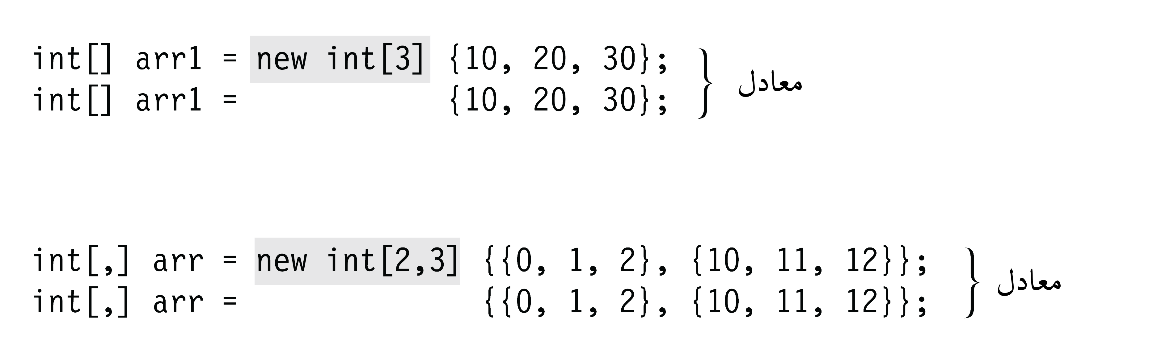
{ {7, 2}, {1, 13}, {9, 3} },

{ {4, 6}, {3, 2}, {23, 8} }

};

دستور میانبر

هنگامی که ایجاد آرایه و مقداردهی اولیه در یک دستور با یک دیگر ترکیب می شوند، می توان بخش دستوری ایجاد آرایه را به طور کامل حذف کرد و تنها بخش مقداردهی اولیه را به کار برد. شکل 9-12 این دستور میانبر را نشان می دهد.



شکل 9-12 میانبری برای تعریف آرایه، ایجاد و مقداردهی اولیه

تعیین ضمنی نوع آرایه ها

تا کنون، به صراحت نوع آرایه ها در ابتدای دستور آرایه مشخص شده است. اما مانند سایر متغییر ها محلی، همچنین می توان نوع آرایه های محلی را به طور ضمنی تعیین کنید. به این معنا که:

* هنگام مقداردهی اولیه آرایه، می توان اجازه دهید که کامپایلر نوع آرایه را از نوع مقادیر اولیه آن استنباط کند. این زمانی مجاز است که تمام مقادیر اولیه بتوانند به یک نوع واحد تبدیل گردند.
* همانند تعیین ضمنی نوع متغییر های محلی، از کلمه کلیدی var به جای نوع آرایه استفاده می شود.

کد زیر نمونه هایی از تعیین ضمنی و صریح نوع سه آرایه را نشان میدهد. مجموعه اول، آرایه ای بعدی از نوع int است. مجموعه دوم، آرایه ای دو بعدی از نوع int است. و مجموعه سوم، آرایه ای از نوع string است. توجه کنید که در تعریف نوع تعیین شده ضمنی intArr4 هنوز هم نیاز است که رده مقادیر اولیه تعیین شوند.

صریح صریح

↓   ↓

int [] intArr1 = new int[] { 10, 20, 30, 40 };

var intArr2 = new [] { 10, 20, 30, 40 };

↑ ↑

کلمه کلیدی استنباط

int[,] intArr3 = new int[,] { { 10, 1 }, { 2, 10 }, { 11, 9 } };

var intArr4 = new [,] { { 10, 1 }, { 2, 10 }, { 11, 9 } };

↑

تعیین کننده رده

string[] sArr1 = new string[] { "life", "liberty", "pursuit of happiness" };

var sArr2 = new [] { "life", "liberty", "pursuit of happiness" };

قرار دادن تمام آنها با یکدیگر

کد زیر تمام تکه هایی که ما تاکنون به آنها نگاه کرده ایم را در کنار یکدیگر قرار می دهد. با استفاده از آرایه مستطیلی شکل آرایه ای ایجاد و آن را مقداردهی اولیه می کند.

// تعریف، ایجاد، و مقداردهی اولیه یک آرایه که نوع آن به طور ضمنی تعیین می شود.

var arr = new int[,] {{0, 1, 2}, {10, 11, 12}};

// چاپ مقادیر آن

for( int i=0; i<2; i++ )

for( int j=0; j<3; j++ )

Console.WriteLine("Element [{0},{1}] is {2}", i, j, arr[i,j]);

این کد خروجی زیر را تولید می کند:

|  |
| --- |
| Element [0,0] is 0  Element [0,1] is 1  Element [0,2] is 2  Element [1,0] is 10  Element [1,1] is 11  Element [1,2] is 12 |

آرایه های دندانه دار

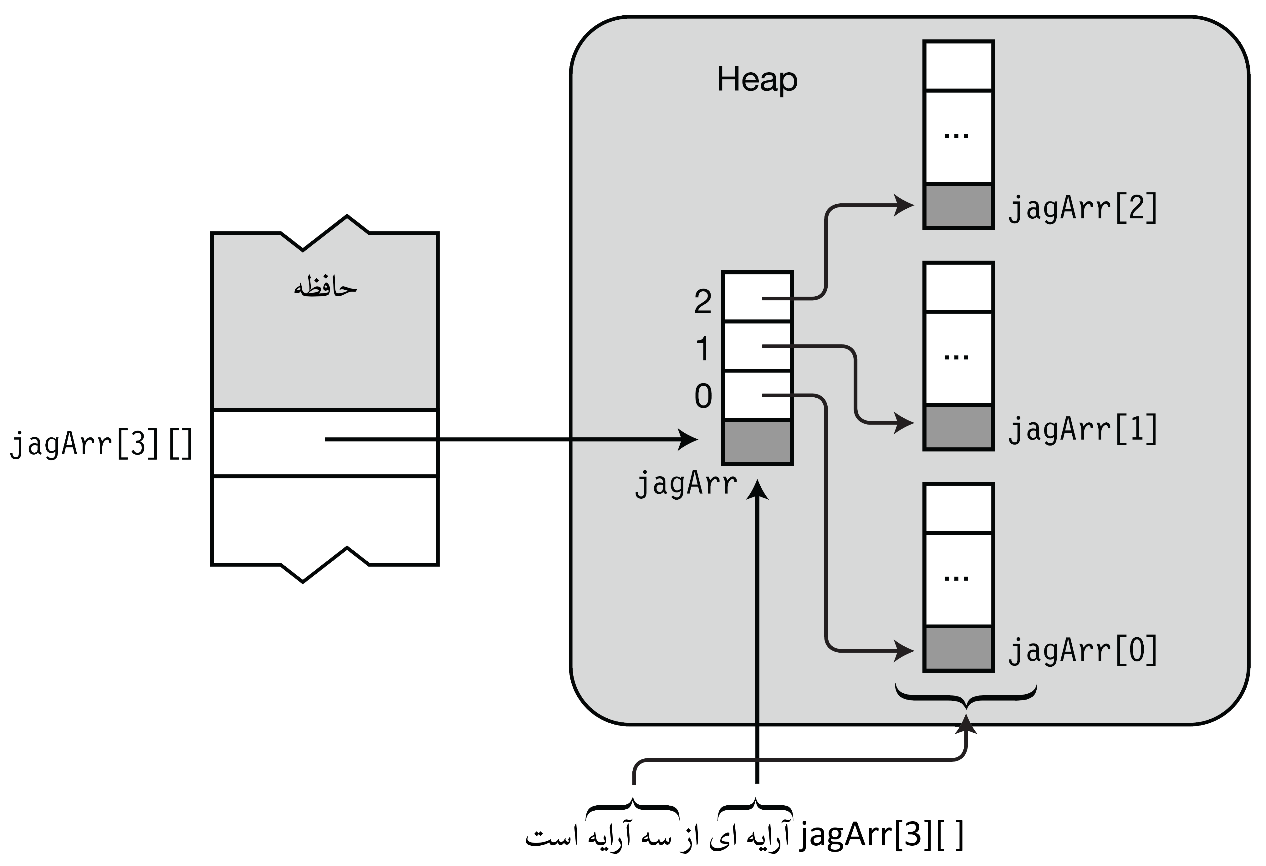
یک آرایه دندانه دار، آرایه ای از آرایه ها است. بر خلاف آرایه های مستطیلی، زیر آرایه های یک آرایه دندانه دار می تواند شماره های مختلفی از عناصر داشته باشد.

به عنوان مثال، کد زیر آرایه ای دندانه دار دو بعدی را تعریف می کند. شکل 10-12 نحو قرارگیری آن را در حافظه نشان می دهد.

* طول بعد اول 3 است.
* این دستور را می توان به عنوان " jagArr آرایه ای است که سه آرایه از نوع int دارد" خواند.
* توجه کنید که شکل چهار آرایه را نشان می دهد. یک آرایه برای آرایه سطح بالا و سه آرایه برای زیر آرایه ها.

int[][] jagArr = new int[3][]; // تعریف و ایجاد آرایه سطح بالا

... // تعریف و ایجاد زیر آرایه ها



شکل 10-12 یک آرایه دندانه دار، آرایه ای از آرایه ها است.

تعریف یک آرایه دندانه دار

در دستور تعریف آرایه های دندانه دار نیاز است که برای هر یک از ابعاد، براکت جداگانه ای قرار داده شود. تعداد براکت ها در تعریف متغییر رده آرایه را تعیین می کند.

* آرایه دندانه دار می تواند هر تعداد بعد که نیاز است را داشته باشد.
* همانند آرایه مستطیلی، طول بعد نمی تواند در قسمت نوع آرایه باشد.

تعیین رده

↓

int[][] SomeArr; // 2 = رده

int[][][] OtherArr; // 3 = رده

↑ ↑

نوع آرایه نوع آرایه

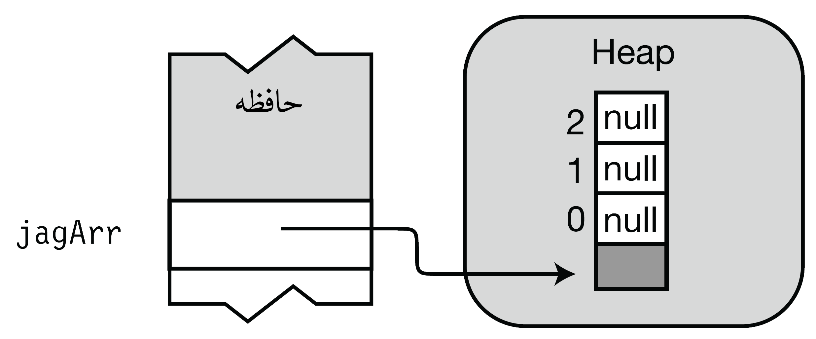
نمونه سازی میانبر

می توان آرایه های دندانه دار را با ایجاد سطح اول ارایه ایجاد کرد مانند دستور ترکیب شده زیر. شکل 11-12 نتیجه کار را نشان می دهد.

سه زیر آرایه

↓

int[][] jagArr = new int[3][];



شکل 11-12 نمونه سازی سطح اول آرایه

نمی توان بیشتر از آرایه سطح اول را در دستور ایجاد کننده آرایه ایجاد کنید.

مجاز

↓

int[][] jagArr = new int[3][4]; // اشتباه! خطای کامپایل

↑

غیر مجاز

مقداردهی اولیه آرایه دندانه دار

برخلاف آرایه های دیگر، نمی توان به طور کامل آرایه دندانه دار را نمونه سازی کرد. از آنجایی که یک آرایه دندانه دار مجموعه ای از آرایه های مستقل است، هر آرایه باید به طور جداگانه ایجاد شود. برای مقدار دهی اولیه و کامل یک آرایه دندانه دار به مراحل زیر نیاز است:

* نمونه سازی آرایه سطح بالا
* نمونه سازی هریک از زیر آرایه ها به طور جداگانه و تخصیص دادن مرجع آرایه ایجاد شده به عنصر آرایه سطح بالا.

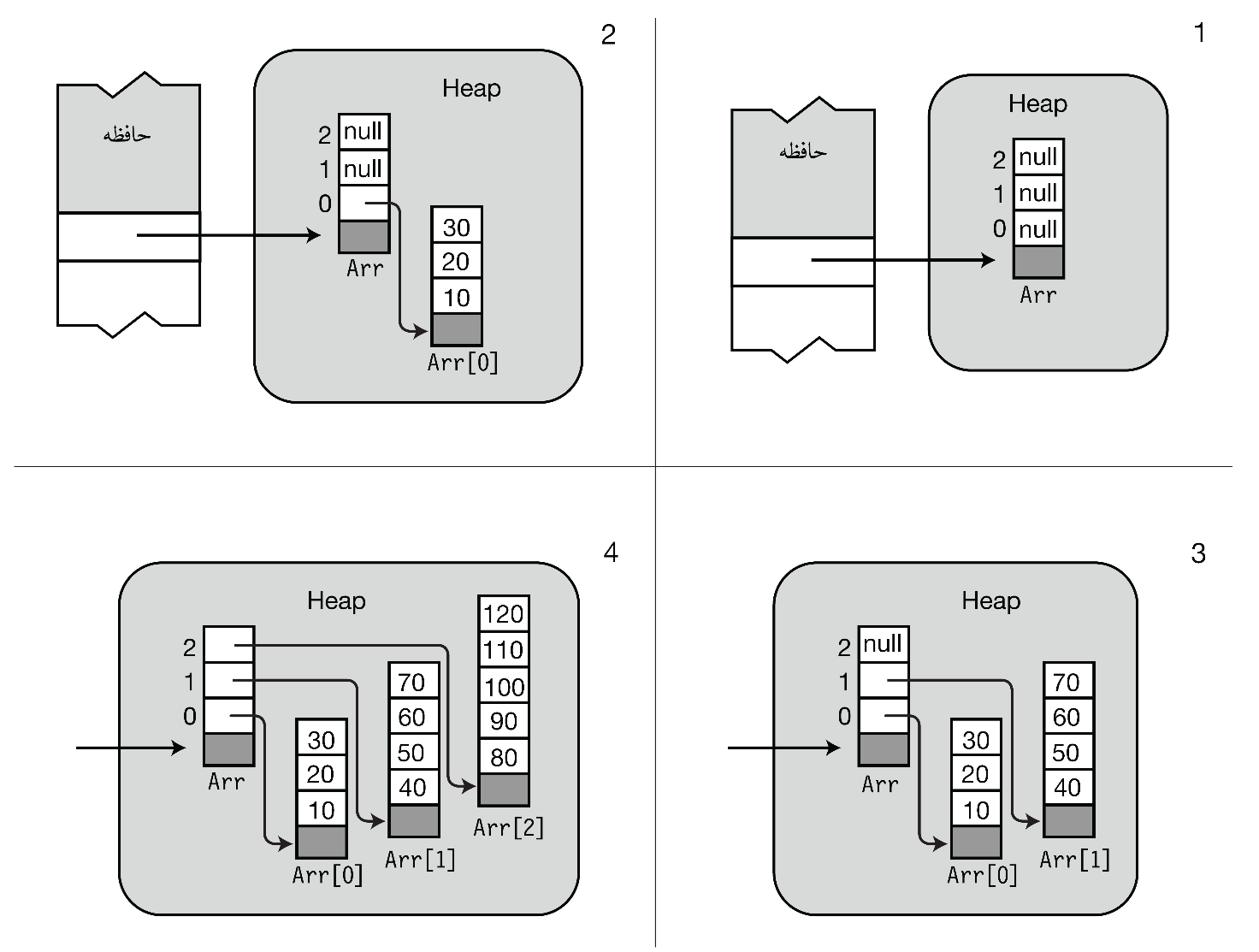
به عنوان مثال، کد زیر تعریف، نمونه سازی و مقداردهی اولیه یک آرایه دندانه دار دو بعدی را نشان می دهد. توجه کنید که مرجع هر زیر آرایه به عنصر آرایه سطح بالا تخصیص داده می شود. مراحل 1 تا 4 موجود در کد با بازنمایی های شکل 12-12 مطابقت دارد.

int[][] Arr = new int[3][]; // 1- نمونه سازی آرایه سطح بالا

Arr[0] = new int[] {10, 20, 30}; // 2- نمونه سازی زیر آرایه

Arr[1] = new int[] {40, 50, 60, 70}; // 3- نمونه سازی زیر آرایه

Arr[2] = new int[] {80, 90, 100, 110, 120}; // 4- نمونه سازی زیر آرایه



شکل 12-12 ایجاد یک آرایه دندانه دار دو بعدی

زیر آرایه ها در آرایه های دندانه دار

از آنجایی که زیر آرایه ها در آرایه های دندانه دار خود نیز آرایه هستند، این امکان وجود دارد که آرایه مستطیلی در داخل آرایه های دندانه دار باشند. به عنوان مثال، کد زیر آرایه ای دندانه دار از سه آرایه مستطیلی دو بعدی ایجاد می کند و آنها را مقداردهی می کند. سپس مقادیر آنها را نمایش می دهد. شکل 13-12 ساختار آن را نشان می دهد.

این کد از متد GetLength(int n) از آرایه که از System.Array ارث برده شده است استفاده می کند تا طول ابعاد مشخص شده از آرایه را بدست آورد.

int[][,] Arr; // آرایه ای از آرایه های دو بعدی

Arr = new int[3][,]; // مقداردهی آرایه با سه آرایه دو بعدی

Arr[0] = new int[,] { { 10, 20 },

{ 100, 200 } };

Arr[1] = new int[,] { { 30, 40, 50 },

{ 300, 400, 500 } };

Arr[2] = new int[,] { { 60, 70, 80, 90 },

{ 600, 700, 800, 900 } };

↓ Arr ،0 تعیین طول بعد

for (int i = 0; i < Arr.GetLength(0); i++)

{

↓ Arr[ i ] ،0 تعیین طول بعد

for (int j = 0; j < Arr[i].GetLength(0); j++)

{

↓ Arr[ i ] ،1 تعیین طول بعد

for (int k = 0; k < Arr[i].GetLength(1); k++)

{

Console.WriteLine

("[{0}][{1},{2}] = {3}", i, j, k, Arr[i][j, k]);

}

Console.WriteLine("");

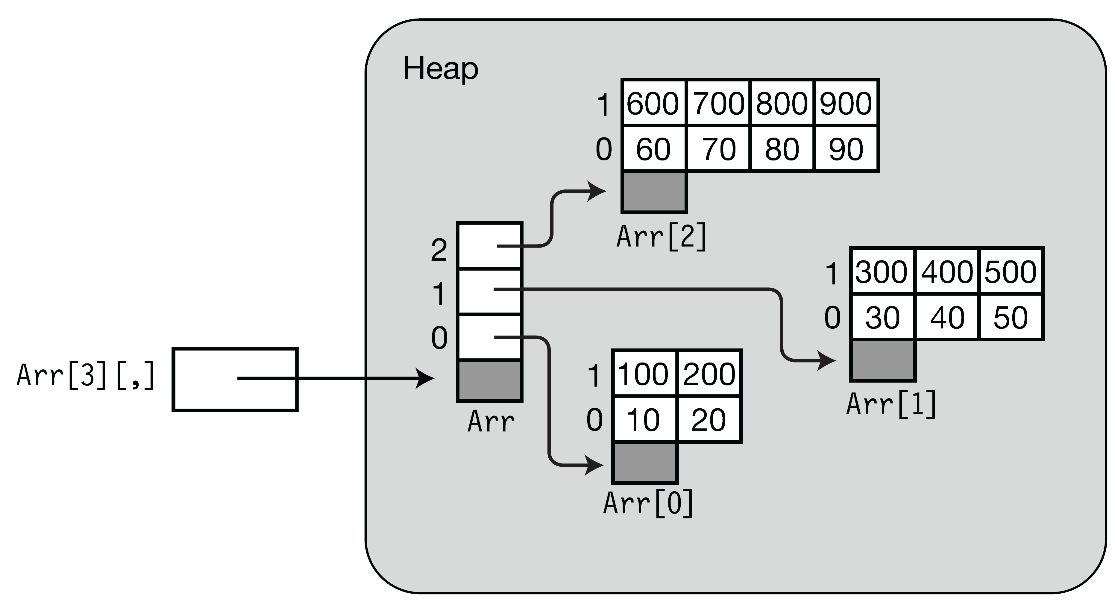
}

Console.WriteLine("");

}

این کد خروجی زیر را تولید می کند:

|  |
| --- |
| [0][1,0] = 100  [0][1,1] = 200  [1][0,0] = 30  [1][0,1] = 40  [1][0,2] = 50  [1][1,0] = 300  [1][1,1] = 400  [1][1,2] = 500  [2][0,0] = 60  [2][0,1] = 70  [2][0,2] = 80  [2][0,3] = 90  [2][1,0] = 600  [2][1,1] = 700  [2][1,2] = 800  [2][1,3] = 900 |

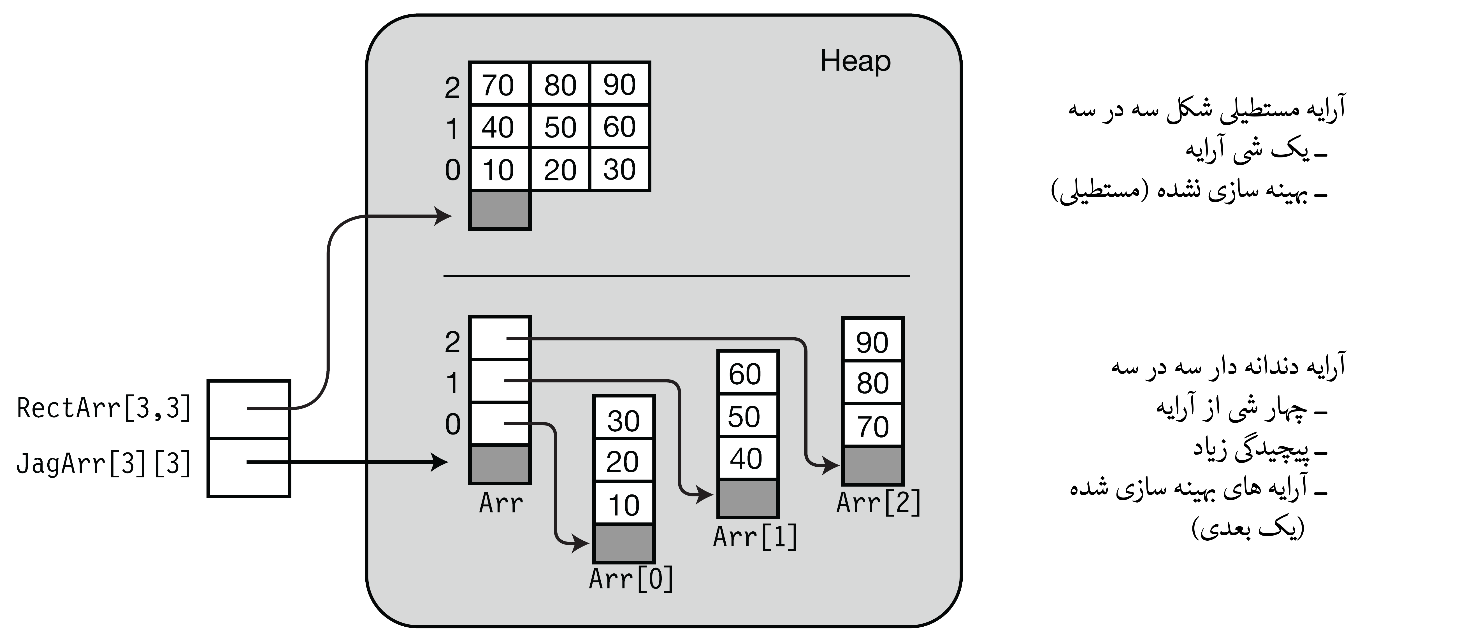


شکل 13-12 آرایه دندانه دار از سه آرایه مستطیلی شکل

مقایسه آرایه های مستطیلی و دندانه دار

ساختار آرایه های مستطیلی و دندانه دار به طور قابل توجهی تفاوت دارند. به عنوان مثال، شکل 14-12 ساختار آرایه مستطیلی سه در سه و همچنین یک آرایه دندانه دار از سه آرایه یک بعدی به طول سه را نشان می دهد.

* هر دو آرایه نه عدد صحیح را نگه داری می کنند، اما همانطور که می بینید ساختاری کاملا متفاوت دارند.
* آرایه مستطیلی یک شی از آرایه دارد، در حالی که آرایه دندانه دار چهار شی از آرایه دارد.



شکل 14-12 مقایسه ساختار آرایه های مستطیلی و دندانه دار

آرایه های یک بعدی دستورالعمل خاصی در CIL دارند که اجازه می دهد آنها عملکرد بهینه ای داشته باشند. آرایه های مستطیلی این دستورالعمل را ندارد و به همان سطح بهینه سازی شده نیستند. به این دلیل، استفاده از آرایه های یک بعدی (که می توانند بهینه سازی شوند) برای آرایه های دندانه دار می تواند کارآمد تر باشد.

از سوی دیگر، پیچیدگی برنامه نویسی می تواند برای یک آرایه مستطیلی به طور قابل توجهی کمتر باشد زیرا به جای آرایه ای از آرایه ها، می تواند به عنوان یک واحد مورد بررسی قرار گیرد.

دستور foreach

دستور foreach اجازه می دهد تا به ترتیب به هر عنصر آرایه دسترسی پیدا شود. این در واقع یک ساختار کلی است که همچنین در آن با دیگر نوع های مجموعه[[9]](#footnote-9) کار می شود اما در این بخش تنها استفاده از آن را با آرایه ها بحث خواهیم کرد. در فصل 18 استفاده از آن را با دیگر نوع های مجموعه بررسی خواهیم کرد.

مواردی که مهم است درباره دستور foreach بدانید به شرح زیر است:

* متغییر تکرار[[10]](#footnote-10) یک متغییر موقت از نوع همان عناصر آرایه است. دستور foreach از متغییر تکرار استفاده می کند تا به ترتیب نماینده ای برای هر یک از عناصر آرایه باشد.
* نحو استفاده از دستور foreach در زیر نشان داده شده است، که
  + Type نوع عناصر آرایه است. می توان به صراحت نوع آن را تعیین کنیم و چون کامپایلر می داند نوع آرایه چیست می توان با استفاده از var اجازه دهید تا به طور ضمنی کامپایلر آن را استنباط کند.
  + Identifier نام متغییر تکرار است.
  + ArrayName نام آرایه است که به ترتیب عناصر آن بررسی می شود.
  + Statement یک دستور ساده یا بلوکی از دستورات است که یک بار برای هر عنصر اجرا می شود.

نوع متغییر تکرار به صراحت تعیین شده

↓

foreach( Type *Identifier* in *ArrayName* )

*Statement*

نوع متغییر تکرار به طور ضمنی تعیین می شود

↓

foreach( var *Identifier* in *ArrayName* )

*Statement*

در ادامه متن، گاهی از نوع ضمنی و گاهی از نوع صریح استفاده خواهیم کرد، به طوریکه می توانید نوع دقیق استفاده شده را مشاهده کنید.

دستور foreach به صورت زیر کار می کند:

* با اولین عنصر آرایه شروع می شود و مقدار این عنصر را به متغییر تکرار تخصیص می دهد.
* سپس بدنه این دستور اجرا می شود. در داخل بدنه دستور، می توان متغییر تکرار را به عنوان نام مستعار فقط خوانی برای عنصر آرایه استفاده کرد.
* بعد از اجرای بدنه، دستور foreach عنصر بعدی آرایه را انتخاب می کند و این فرآیند را تکرار می کند.

این روش، اجازه می دهد به هر یک از عناصر آرایه، یک به یک دسترسی داشته باشید. به عنوان مثال، کد زیر نحو استفاده از دستور foreach را با یک آرایه یک بعدی نشان می دهد:

* دستور WriteLine ، که در بدنه دستور foreach قرار دارد، یک بار برای هر یک از عناصر آرایه اجرا می شود.
* در حلقه، اولین بار متغییر تکرار مقدار اولین عنصر آرایه را دارد. هر بار به طور پی در پی متغییر تکرار، مقدار عنصر بعدی آرایه را دارد.

int[] arr1 = {10, 11, 12, 13};

تعریف متغییر تکرار

↓   استفاده از متغییر تکرار

foreach( int item in arr1 ) ↓

Console.WriteLine("Item Value: {0}", item);

این کد خروجی زیر را تولید می کند:

|  |
| --- |
| Item Value: 10  Item Value: 11  Item Value: 12  Item Value: 13 |

متغییر تکرار فقط خوانی است

به دلیل اینکه مقدار متغییر تکرار فقط خوانی است، به صراحت نمی توان آن را تغییر داد. اما این بر روی آرایه های نوع مقداری و آرایه های نوع ارجاعی تاثیر متفاوتی می گذارد.

برای آرایه های نوع مقداری، این بدان معناست که توسط متغییر تکرار نمی توان یک عنصر از آرایه را تغییر داد. به عنوان مثال، در کد زیر، تلاش برای تغییر متغییر تکرار در زمان کامپایل یک پیغام خطا تولید می کند.

int[] arr1 = {10, 11, 12, 13};

foreach( int item in arr1 )

item++; // خطای کامپایل. تغییر مقدار متغییر مجاز نیست

برای آرایه های نوع مقداری، هنوز هم نمی توان متغییر تکرار را تغییر داد، اما متغییر تکرار تنها شامل مرجع داده ها است و داده های خود را نگه داری نمی کند. بنابراین اگر چه نمی توان مرجع را تغییر داد اما می توان داده ها را از طریق متغییر تکرار تغییر داد.

کد زیر آرایه ای از چهار شی MyClass ایجاد می کند و آنها را مقداردهی می کند. در اولین دستور foreach ، داده های هر یک از شی ها تغییر می کند. در دومین دستور foreach ، داده های تغییر داده شده از شی ها خوانده می شوند.

class MyClass

{

public int MyField = 0;

}

class Program

{

static void Main()

{

MyClass[] mcArray = new MyClass[4]; // ایجاد آرایه

for (int i = 0; i < 4; i++)

{

mcArray[i] = new MyClass(); // ایجاد شی

mcArray[i].MyField = i; // تعیین مقدار فیلد

}

foreach (MyClass item in mcArray)

item.MyField += 10; // تغییر داده

foreach (MyClass item in mcArray)

Console.WriteLine("{0}", item.MyField); // خواندن داده تغییر داده شده

}

}

این کد خروجی زیر را تولید می کند:

|  |
| --- |
| 10  11  12  13 |

دستور foreach با آرایه های چند بعدی

در یک آرایه چند بعدی، عناصر به ترتیبی که شاخص سمت راست سریع تر افزایش می یابد پردازش می شوند. زمانی که شاخص از 0 تا length – 1 رفته است، شاخص بعدی (سمت چپ) یک واحد افزایش می یابد و شاخص سمت راست مجدد به 0 تغییر پیدا می کند.

مثالی با یک آرایه مستطیلی

مثال زیر نحو استفاده از دستور foreach را با یک آرایه مستطیلی نشان می دهد:

class Program

{

static void Main()

{

int total = 0;

int[,] arr1 = { {10, 11}, {12, 13} };

foreach( var element in arr1 )

{

total += element;

Console.WriteLine

("Element: {0}, Current Total: {1}", element, total);

}

}

}

این کد خروجی زیر را تولید می کند:

|  |
| --- |
| Element: 10, Current Total: 10  Element: 11, Current Total: 21  Element: 12, Current Total: 33  Element: 13, Current Total: 46 |

مثالی با یک آرایه دندانه دار

از آنجایی که آرایه های دندانه دار، آرایه ای از آرایه ها است، باید دستورات حلقه foreach جداگانه ای برای هر یک از ابعاد در آرایه دندانه دار استفاده کرد. دستورات foreach باید به طور صحیح تودرتو شده باشند تا مطمئن شوید که هر یک از آرایه های تودرتو به درستی پردازش می شوند.

به عنوان مثال، در کد زیر، اولین دستور foreach از آرایه سطح بالای arr1 استفاده می کند تا زیر آرایه ها را پردازش کند. دومین دستور foreach عناصر زیر آرایه در آرایه سطح بالا را پردازش می کند.

class Program

{

static void Main( )

{

int total = 0;

int[][] arr1 = new int[2][];

arr1[0] = new int[] { 10, 11 };

arr1[1] = new int[] { 12, 13, 14 };

foreach (int[] array in arr1) // پردازش آرایه سطح بالا

{

Console.WriteLine("Starting new array");

foreach (int item in array) // پردازش دومین سطح از آرایه

{

total += item;

Console.WriteLine

(" Item: {0}, Current Total: {1}", item, total);

}

}

}

}

این کد خروجی زیر را تولید می کند:

|  |
| --- |
| Starting new array  Item: 10, Current Total: 10  Item: 11, Current Total: 21  Starting new array  Item: 12, Current Total: 33  Item: 13, Current Total: 46  Item: 14, Current Total: 60 |

کواریانس آرایه

تحت شرایط خاصی می توان شی ای به یک عنصر از آرایه اختصاص داد حتی اگر شی از نوع آرایه نباشد. این ویژگی از آرایه ها کواریانس آرایه نامیده می شود. اگر موارد زیر صادق باشد می توان از کواریانس آرایه استفاده کرد:

* آرایه یک آرایه نوع ارجاعی باشد.
* بین نوع تخصیص داده شونده و نوع آرایه بتوان تبدیل ضمنی یا صریح انجام داد.

از آنجایی که بین کلاس مشتق شده و کلاس پایه همیشه تبدیل ضمنی وجود دارد، می توان یک شی از کلاس مشتق شده را یک آرایه تعریف شده از کلاس پایه اختصاص داد.

به عنوان مثال، کد زیر دو کلاس A و B را تعریف می کند، که در آن کلاس B از کلاس A مشتق شده است. خط آخر با اختصاص دادن شی ای از نوع B به عناصر آرایه (که از نوع A تعریف شده)، کواریانس را نشان می دهد. شکل 15-12 نحو قرار گیری آن را در حافظه نشان می دهد.

class A { ... } // کلاس پایه

class B : A { ... } // کلاس مشتق شده

class Program {

static void Main() {

// A[] دو آرایه از نوع

A[] AArray1 = new A[3];

A[] AArray2 = new A[3];

// A به آرایه ای از نوع A تخصیص عادی شی ای از نوع

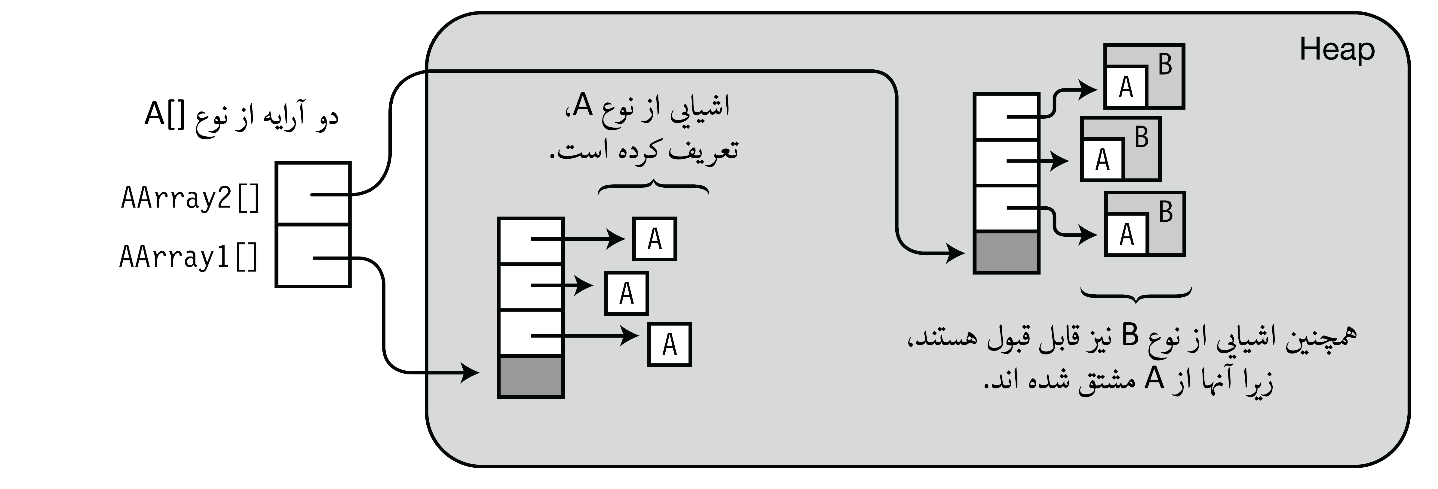
AArray1[0] = new A(); AArray1[1] = new A(); AArray1[2] = new A();

// A به آرایه ای از نوع B تخصیص کواریانس شی از نوع

AArray2[0] = new B(); AArray2[1] = new B(); AArray2[2] = new B();

}

}



شکل 15-12 نمایش کواریانس آرایه ها

|  |
| --- |
| * توجه هیچ کواریانسی برای آرایه های نوع مقداری وجود ندارد. |

عضو هایی مفید از ارث بری آرایه ها

قبلا هم اشاره شد که آرایه های C# از کلاس System.Array مشتق شده است. از این کلاس پایه تعدادی از خصوصیات و متدهای مفید به ارث می رسد. جدول 1-12 برخی از آنها که بیشترین استفاده را دارند لیست کرده است.

جدول 1-12 برخی از عضو های مفید ارث برده شده توسط آرایه ها

|  |  |  |  |
| --- | --- | --- | --- |
| عضو | نوع | محدوده استفاده | معنی |
| Rank | خصوصیت | نمونه | تعداد ابعاد آرایه را بدست می آورد |
| Length | خصوصیت | نمونه | تعداد کل عناصر در تمام ابعاد آرایه را بدست می آورد |
| GetLength | متد | نمونه | طول ابعاد خاصی از آرایه را برمی گرداند |
| Clear | متد | استاتیک | محدوده ای از عناصر با 0 یا null مقداردهی می کند |
| Sort | متد | استاتیک | عناصر آرایه یک بعدی را مرتب می کند |
| BinarySearch | متد | استاتیک | مقداری را در یک آرایه یک بعدی با استفاده از جستجوی دودویی، جستجو می کند |
| Clone | متد | نمونه | یک کپی سطحی از آرایه را انجام می دهد برای هر دو آرایه های نوع مقداری و آرایه های نوع ارجاعی، تنها عناصر را کپی می کند |
| IndexOf | متد | استاتیک | شاخص اولین رخداد مقداری در یک آرایه یک بعدی را برمی گرداند |
| Reverse | متد | استاتیک | طیف وسیعی از عناصر یک آرایه یک بعدی را معکوس می کند |
| GetUpperBound | متد | نمونه | کران بالا در بعد مشخص شده را بدست می آورد |

به عنوان مثال، کد زیر از برخی خصوصیات و متد ها استفاده می کند:

public static void PrintArray(int[] a)

{

foreach (var x in a)

Console.Write("{0} ", x);

Console.WriteLine("");

}

static void Main()

{

int[] arr = new int[] { 15, 20, 5, 25, 10 };

PrintArray(arr);

Array.Sort(arr);

PrintArray(arr);

Array.Reverse(arr);

PrintArray(arr);

Console.WriteLine();

Console.WriteLine("Rank = {0}, Length = {1}",arr.Rank, arr.Length);

Console.WriteLine("GetLength(0) = {0}",arr.GetLength(0));

Console.WriteLine("GetType() = {0}",arr.GetType());

}

این کد خروجی زیر را تولید می کند:

|  |
| --- |
| 15 20 5 25 10  5 10 15 20 25  25 20 15 10 5  Rank = 1, Length = 5  GetLength(0) = 5  GetType() = System.Int32[] |

متد Clone

متد clone یک کپی سطحی از آرایه را انجام می دهد. این بدان معنی است که تنها یک کلون از آرایه خود را ایجاد می کند. اگر یک آرایه نوع ارجاعی باشد، شی های اشاره شده توسط عناصر کپی نمی شوند. این متد برای آرایه های نوع مقداری و آرایه های نوع ارجاعی نتایج متفاوتی دارد.

* از شبیه سازی یک آرایه نوع مقداری دو آرایه مستقل نتیجه می شود.
* از شبیه سازی یک آرایه نوع ارجاعی دو آرایه نتیجه می شود که عناصر این دو آرایه به شی های یکسانی اشاره می کنند.

متد Clone مرجعی از نوع شی برمی گرداند، که باید به نوع آرایه تبدیل شود.

int[] intArr1 = { 1, 2, 3 };

نوع آرایه یک شی برمی گرداند

↓   ↓

int[] intArr2 = ( int[] ) intArr1.Clone();

به عنوان مثال، کد زیر مثالی از شبیه سازی آرایه نوع مقداری را نشان می دهد، که دو آرایه مستقل تولید می کند. شکل 16-12 مراحل نمایش داده شده در کد را نشان می دهد.

static void Main()

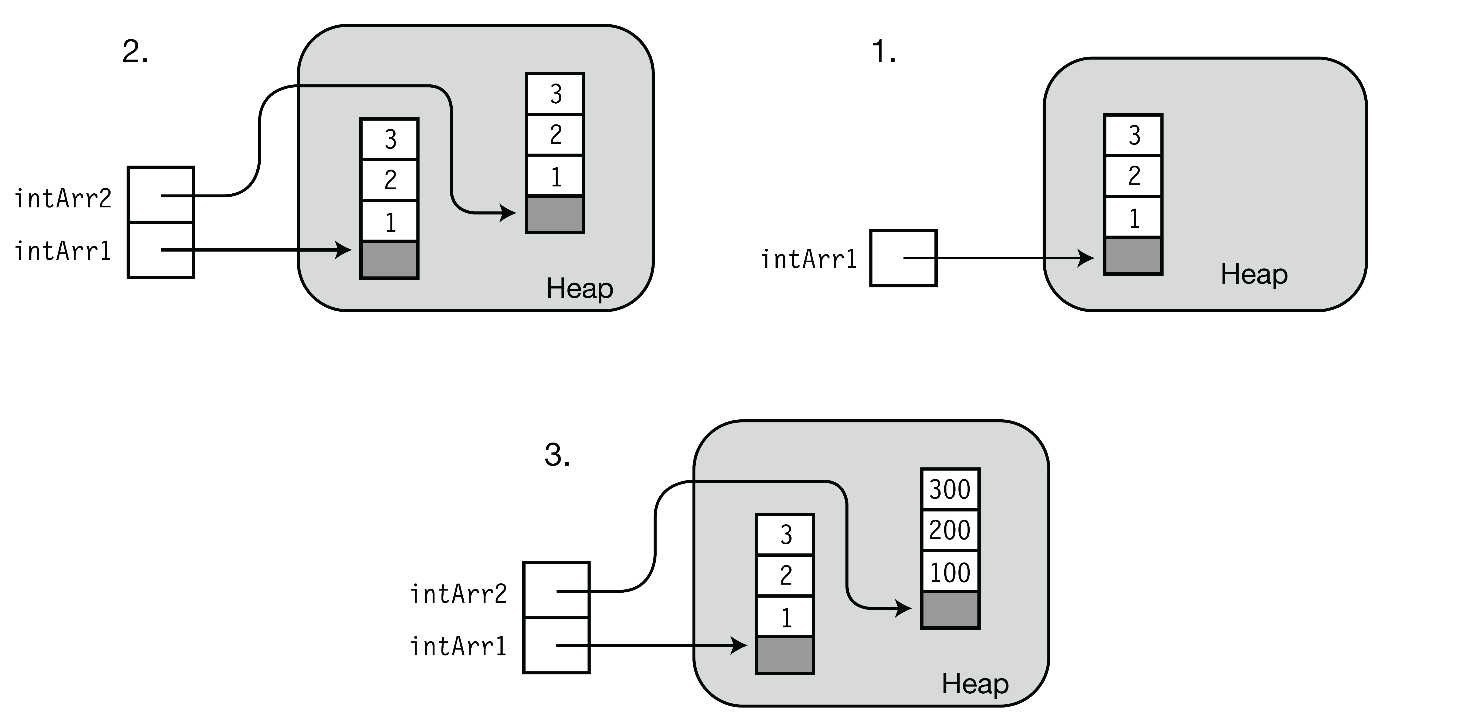
{

int[] intArr1 = { 1, 2, 3 }; // مرحله 1

int[] intArr2 = (int[]) intArr1.Clone(); // مرحله 2

intArr2[0] = 100; intArr2[1] = 200; intArr2[2] = 300; // مرحله 3

}



شکل 16-12 شبیه سازی یک آرایه نوع مقداری دو آرایه مستقل تولید می کند.

از شبیه سازی یک آرایه نوع ارجاعی دو آرایه نتیجه می شود که عناصر این دو آرایه به شی های یکسانی اشاره می کنند. کد زیر مثالی از شبیه سازی آرایه های نوع ارجاعی نشان می دهد. شکل 17-12 مراحل نمایش داده شده در کد را نشان می دهد.

class A

{

public int Value = 5;

}

class Program

{

static void Main()

{

A[] AArray1 = new A[3] { new A(), new A(), new A() }; // مرحله 1

A[] AArray2 = (A[]) AArray1.Clone(); // مرحله 2

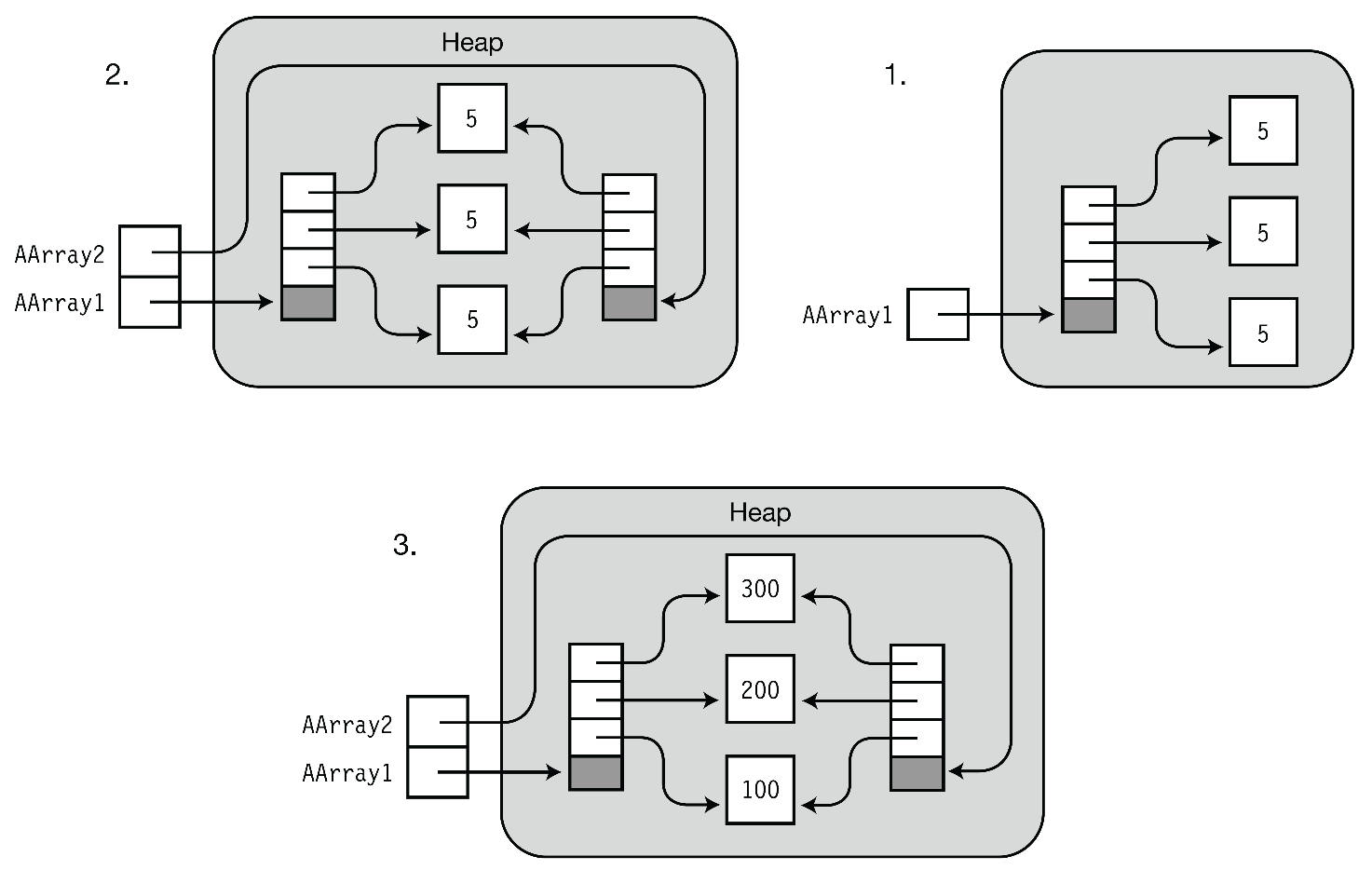
AArray2[0].Value = 100;

AArray2[1].Value = 200;

AArray2[2].Value = 300; // مرحله 3

}

}



شکل 17-12 شبیه سازی یک آرایه نوع ارجاعی دو آرایه ارجاعی از همان شی ها تولید می کند.

مقایسه انواع آرایه

جدول 2-12 برخی از شباهت ها و تفاوت های بین سه نوع از آرایه ها را خلاصه کرده است.

جدول 2-12 مقایسه انواع آرایه ها

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| نوع آرایه | تعداد شی های آرایه | نحو | | شکل |
| کروشه | کاما |
| آرایه یک بعدی   * دارای دستورالعملی بهینه در CIL | یکی | یک جفت | ندارد | H:\Programmer\Translate\chapter 12\Image\tab 2.12.1.png |
| مستطیلی   * چند بعدی * همه زیر آرایه ها در آرایه مستطیلی باید یک اندازه باشند. | یکی | یک جفت | دارد | H:\Programmer\Translate\chapter 12\Image\tab 2.12.2.png |
| دندانه دار   * چند بعدی * زیر آرایه ها می توانند طول های مختلفی داشته باشند | چندین | چندین جفت | ندارد | H:\Programmer\Translate\chapter 12\Image\tab 2.12.3.png |

1. Elements [↑](#footnote-ref-1)
2. Rank [↑](#footnote-ref-2)
3. Dimensions [↑](#footnote-ref-3)
4. Dimension length [↑](#footnote-ref-4)
5. Array length [↑](#footnote-ref-5)
6. Rectangular array [↑](#footnote-ref-6)
7. Jagged array [↑](#footnote-ref-7)
8. Rank specifies [↑](#footnote-ref-8)
9. Collection type [↑](#footnote-ref-9)
10. Iteration variable [↑](#footnote-ref-10)