آرایه ها و رشته ها

این صفحه نحوه تعریف و کار با آرایه های یک بعدی، چندبعدی، کاراکتری را مشاهده خواهید کرد. آرایه ها روشی برای ذخیره متغیرهای هم نوع تحت یک نام است. رشته به عنوان نوعی آرایه کاراکتری درنظر گرفته می شود. توابع آماده ای برای کار با رشته ها در C++ وجود دارد.

آرایه
مقداردهی اولیه آرایه
بیشترین اندازه آرایه
آرایه های کاراکتری (رشته)
نوع داده string
توابع رشته ای

در بعضی حالات داده های مرتبطی دارید که بطور منطقی تحت یک نام می توانند گروه بندی شوند. مثلا اگر بخواهید حرارت ۲۰ روز متوالی را ذخیره کنید کلیه مقادیر منطقا با نام tempreture باید اسمگذاری شوند. برای ذخیره چند مقدار با یک نام آرایه روش مناسبی است.

آرایه

یک آرایه (array) مجموعه ای از محل های پشت سرهم حافظه است که همگی دارای یک نام می باشند . یک آرایه نوعی متغیر است که بجای ذخیره یک مقدار یکسری از مقادیر هم نوع را ذخیره می کند. هر محل ذخیره سازی در آرایه را یک عنصر آرایه می نامند.

شکل کلی تعریف آرایه به صورت زیر است:

```
DataType array_name[size];
```

Size تعداد عناصر آرایه است که در کروشه ذکر می شود. DataType نوع عناصر آرایه است که از هر نوع داده ای می تواند باشد.

اندازه آرایه باید در زمان کامپایل مشخص باشد و در طول زمان اجرا ثابت است و تغییر نمی کند. وقتی آرایه ای اعلان می شود کامپایلر یک بلاک از حافظه که برای نگهداری کل آرایه کفایت می کند را کنار می گذارد. به این ترتیب عناصر آرایه پشت سرهم ذخیره می شوند.

اسم آرایه اشاره گری به اولین عنصر آن است. به عناصر می توان به طور منفرد با اضافه کردن اندیس به اسم آرایه مراجعه کرد. اندیس (index) باعث متمایز شدن عناصر آرایه از یکدیگر می شود و تعیین می کند عنصر آرایه چندمین محل ذخیره سازی در آرایه است. در C++ اندیس آرایه یک عدد صحیح است که از صفر شروع می شود.

مثال. آرایه A که به صورت زیر تعریف شده است یک آرایه یک بعدی با ۵ عنصر از نوع صحیح است. عناصر آن شامل A[0]، A[1]، A[2]، A[3] و A[4] است.

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

هنگام دسترسی به عناصر آرایه بعد از اسم آرایه باید درون کروشه شماره اندیس عنصر مورد نظر ذکر شود. عناصر آرایه را می توان توسط دستور انتساب مقداردهی کرد یا مقدار آن را از ورودی دریافت کرد.

```
A[0] = 10;

cout << "Enter a number:";
cin >> A[1];
```

با توجه به اینکه تعداد عناصر آرایه معین است برای کارکردن روی کلیه عناصر آرایه حلقه های for روش مناسبی هستند.

مثال. در برنامه زیر کلیه عناصر یک آرایه از ورودی دریافت و سپس نمایش داده می شود.

```
#include <iostream.h>

int main() {
int A[10];

for (int k = 0; k < 10; k++) {
   cout << "Enter an integer: ";
   cin >> A[k];
   }

for (int k = 0; k < 10; k++)
   cout << A[k] << endl;

  return 0;
}
```

بخاطر داشته باشید اندیس عناصر آرایه با n عنصر از 0 تا n-1 است. اگر به عنصری خارج از این محدوده دسترسی پیدا کنید کامپایلر خطائی نمی گیرد ولی این می تواند به مشکلات جدی منجر بشود چون ممکن است روی داده برنامه های دیگر درون حافظه تاثیر بگذارد.

مثال. در برنامه زیر سعی شده به عنصری خارج از اندازه آرایه دسترسی شود. دستور myarray[10] باعث می شود مقدار ۹۹ در محلی ذخیره شود که ۶ خانه بعد از آرایه myarray است این حافظه ممکن است شامل داده لازم برای برنامه دیگر باشد.

```
#include <iostream.h>

int main() {
  int myarray[5];
  myarray[10] = 99;
  cout << myarray[10] << endl;
  return 0;
}
```

اگر اندیس آرایه اعشاری باشد کامپایلر آن را به یک عدد صحیح گرد می کند.

آرایه های یک بعدی

آرایه یک بعدی (1-dimensional array) برای نگهداری لیستی از مقادیر استفاده می شود. هر عنصر آرایه یک بعدی از طریق یک اندیس مشخص می شود.

مثال. آرایه زیر ۲۰ مکان پشت سرهم حافظه را برای مقادیر ممیز شناور اختصاص می دهد. اولین مکان با tempreture[0] بعدی با tempreture[1] و.... . آخرین عنصر tempreture[19] است.

```
float tempreture[20];
```

آرایه های چند بعدی

می توان بعدهای بیشتری به آرایه داد. در یک آرایه چند بعدی (multidimensional) به بیش از یک عدد برای دسترسی به هر عنصر آرایه نیاز است. یک آرایه دو بعدی به ۲ اندیس و یک آرایه سه بعدی به ۳ اندیس نیاز دارد. محدودیتی برای تعداد ابعاد آرایه در C++ وجود ندارد اما بندرت آرایه بیشتر از دو یا سه بعد دیده شده است.

مثال. آرایه زیر دارای دو بعد است که اندازه هر بعد آن ۴ است. بنابراین آرایه ۱۶=۴×۴ عدد صحیح را نگه می دارد. به هر عدد از طریق دو اندیس دسترسی می شود. مثلا عنصر اول myarray[0][0] است.

```
 int myarray[4][4];
```

آرایه های دو بعدی دارای ساختار جدولی هستند و به صورت مجموعه ای از سطرها و ستون ها دیده می شوند.

جدول زیر موقعیت عناصر مثال قبل که دارای ۴ سطر و ۴ ستون است را نشان می دهد.

0,0	*0,1*	*0,2*	*0,3*
1,0	*1,1*	*1,2*	*1,3*
2,0	*2,1*	*2,2*	*2,3*
3,0	*3,1*	*3,2*	*3,3*

عناصر آرایه صرفنظر از تعداد بعدهای آن سطر به سطر در خانه های پشت سرهم حافظه ذخیره می شوند. برای مطالعه بیشتر درباره روش سطری می توانید به بخش نمایش آرایه در درس ساختمان داده مراجعه نمایید.

مقداردهی اولیه آرایه

مانند یک متغیر هنگام اعلان یک آرایه مقدار عناصر آنرا می توان تعیین کرد. آرایه های سراسری به طور پیش فرض توسط صفر مقداردهی می شوند. اما وقتی یک آرایه به صورت محلی تعریف می شود محتوایش نامعین است. بنابراین بهتر است آنرا مقداردهی اولیه کرد.

برای انجام این کار در خط اعلان به دنبال اسم آرایه لیست مقادیر عناصر آن به ترتیب درون یک آکولاد نوشته می شود.

مثال. دستور زیر مقدار 100 را به array[0]، 200 را به array[1]، 300 را به array[2] و 400 را به array[3] اختصاص می دهد.

```
int array[4] = { 100, 200, 300, 400 };
```

اگر اندازه آرایه ذکر نشود کامپایلر اندازه کافی را برای ذخیره لیست مقادیر ذکر شده درنظر می گیرد.

مثال. عبارت زیر مشابه مثال قبل عمل می کند. اندازه آرایه توسط کامپایلر ۴ درنظرگرفته می شود.

```
int array[] = { 100, 200, 300, 400 };
```

اگر تعداد مقادیر درون آکولاد کمتر از تعداد عناصر آرایه باشد، بقیه عناصر بطورخودکار صفر درنظر گرفته می شوند. اگر مقادیر بیشتر از اندازه آرایه باشد کامپایلر خطا می گیرد.

مثال. عناصر array[3] تا array[9] با صفر مقداردهی می شوند.

```
int array[10] = { 1, 2, 3 };
```

در آرایه های چندبعدی لیست مقادیر به ترتیب سطری در عناصر قرار می گیرند.

مثال. در آرایه دو بعدی زیر

```
int array[4][3] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
```

مقادیر به صورت زیر ذخیره می شوند:

array[0][0] برابر با 1
array[0][1] برابر با 2
array[0][2] برابر با 3
array[1][0] برابر با 4
array[1][1] برابر با 5
array[1][2] برابر با 6
...
array[3][1] برابر با 11
array[3][2] برابر با 12

برای خوانائی بیشتر مقادیر هر سطر را می توان درون یک جفت آکولاد دسته بندی کرد.

مثال. نتیجه دستورزیر مشابه مثال قبل است.

```
int array[4][3] = { { 1, 2, 3 } ,
                    { 4, 5, 6 } ,
                    { 7, 8, 9 } ,
                    { 10, 11, 12 } };
```

بیشترین اندازه آرایه

به دلیل مدلی که حافظه کار می کند متغیری با بیشتر از 64KB فضای اشغالی نمی توان داشت. اگر هیچ متغیر دیگری در برنامه وجود نداشته باشد یک آرایه یک بعدی حداکثر می تواند 64KB فضا اشغال کند. البته بعضی از سیستم عامل ها این محدودیت را ندارند.

فضای مورد نیاز برای ذخیره آرایه به تعداد کل عناصر و تعداد بایت های هر عنصر بستگی دارد. برای محاسبه فضای یک آرایه تعداد عناصر آن در تعداد بایت های نوع داده عناصر ضرب می شود.

مثال. میزان فضائی که یک آرایه با ۵۰۰ عنصر از نوع float اشغال می کند برابر با 500×4=2000 بایت است.

نکته. ارسال آرایه به توابع همیشه به صورت مرجع است.

آرایه های کاراکتری (رشته)

یکی از کاربرد های گسترده آرایه به صورت آرایه کاراکتری است. یک آرایه کاراکتری را به عنوان رشته ای از کاراکترها می توان در نظر گرفت. رشته (string) به مجموعه ای از کاراکترها گفته می شود که برای ذخیره متن استفاده می شود.

مثال. دستور زیر آرایه کاراکتری با اندازه ۱۰ ایجاد می کند. این آرایه می تواند رشته ای با حداکثر طول ۹ را ذخیره کند.

```
char UserName[10];
```

C++ انتهای یک رشته را با کاراکتر null می شناسد. کاراکتر null کاراکتر با کد اسکی صفر است که به صورت '\0' نوشته می شود. بنابراین یک محل از آرایه کاراکتری برای ذخیره کاراکتر null باید کنار گذاشته شود.

مثال. برای ذخیره یک رشته ۱۰ کاراکتری نیاز به آرایه کاراکتری با اندازه ۱۱ کاراکتر است.

چون رشته در آرایه به صورت کاراکتری ذخیره می شود که به null ختم می شود تمام چیزی که برای تعریف یک رشته نیاز است اشاره گری به ابتدای آن است. همانطور که گفته شد اسم آرایه اشاره گری به اولین عنصر آن است. بنابراین تنها به اسم آرایه برای دسترسی به رشته ای که درون آرایه ذخیره شده نیاز است. کتابخانه استاندارد C++ شامل توابعی برای کارکردن با رشته ها است. برای ارسال رشته ای به این توابع باید از نام آرایه استفاده شود. برای دریافت و نمایش متغیرهای رشته ای توسط دستورات cin و cout هم به همین صورت می توان عمل کرد.

```
cin >> UserName;
cout << UserName;
```

البته خواندن رشته به صورت فوق روش خیلی خوبی نیست زیرا اگر متن ورودی شامل فضای خالی باشد آنرا بعنوان جداکننده مقادیر می شناسد. علاوه بر این اگر طول رشته وارد شده بیشتر از فضای درنظر گرفته شده در متغیر رشته ای باشد باعث بروز مشکل در برنامه می شود.

روش دیگر برای دریافت یک رشته از ورودی استفاده از متد getline شیء cin است. در این حالت متن ورودی می تواند شامل فضاهای خالی هم باشد.

مثال. برنامه زیر متنی با حداکثر طول ۲۴ کاراکتر را از ورودی دریافت می کند و به متغیر name اختصاص می دهد. کاراکتر NULL در انتها اضافه می شود.

```
#include <iostream.h>

int main() {
  char name[25];
  cout << "Please enter your name \n";
  cin.getline(name,25);
  cout << "Hello " << name << endl;
  return 0;
}
```

مقداردهی اولیه آرایه کاراکتری

مشابه آرایه های دیگر آرایه های کاراکتری را می توان هنگام اعلان مقداردهی اولیه کرد. می توان کاراکترها را مجزا درون آکولاد نوشت یا از یک ثابت رشته ای که درون علامت (") قرار دارد استفاده کرد. در حالت دوم کامپایلر اتوماتیک کاراکتر null را در انتهای رشته درج می کند.

```
char string[10] = { 'h', 'e', 'l', 'l', 'o', '\0' };
char string[10] = "hello";
```

وقتی آرایه توسط رشته "hello" مقداردهی می شود در حافظه به صورت زیر ذخیره می شود:

h	e	l	l	o	NULL				
0	1	2	3	4	5	6	7	8	9

اگر اندازه آرایه ذکر نشود کامپایلر اندازه آرایه را با توجه به تعداد کاراکترهای رشته ای محاسبه می کند. در اعلان زیر آرایه با اندازه ۶ کاراکتر مقداردهی می شود.

```
char string[] = "hello";
```

توابع C++ که روی رشته ها کار می کنند طول رشته را با پیدا کردن کاراکتر null محاسبه می کنند. یکی از مشکلاتی است که ممکن است در برنامه بروز کند. این است که کاراکتر null گم بشود در نتیجه برنامه رشته را تا رسیدن به کاراکتر با کد اسکی صفر در حافظه گسترش می دهد.

نوع داده string

برای ذخیره رشته ها غیر از آرایه کاراکتری انتخاب های دیگری هم وجود دارد. C++ شیء string را برای متغیر های رشته ای دارد که برای استفاده باید فایل هدر <string.h> ضمیمه شود. string متدها و خواص خود را دارد که می توانید استفاده کنید. خاصیت طول یکی از خواص string است.

مثال.

```
#include <iostream.h>
#include <string.h>

int main() {
  string stringA = "C++",
  string stringB = "Is Cool",

  cout << "Length of stringA = " << stringA.length() << endl;
  cout << "Length of stringB = " << stringB.length() << endl;
  return 0;
}
```

روش دیگر برای کارکردن با رشته ها استفاده از اشاره گرهاست. برای این منظور کافی است فضائی در حافظه درنظر گرفته شده رشته ای که به null ختم می شود را در آن ذخیره کرد. اشاره گری به اولین کاراکتر این رشته به عنوان شروع رشته مشابه وقتی که رشته درون آرایه ذخیره می شود کار می کند.

مثال. وقتی دستور زیر اجرا می شود رشته جائی در حافظه ذخیره می شود و آدرس شروع آن در اشاره گر message قرار می گیرد.

```
char *message = "Great Caesar\'s Ghost!";
```

اعلان فوق مشابه تعریف زیر است:

```
char message[] = "Great Caesar\'s Ghost!";
```

توابع رشته ای

C++ شامل کتابخانه ای از توابع رشته ای است که برخی از آنها در جدول زیر آمده است. برای استفاده از این توابع فایل string.h باید ضمیمه شده باشد.

تابع	شرح	الگوی فراخوانی
strcpy	کپی یک رشته در دیگری	char *strcpy(char *dest, char *src);
strlen	برگرداندن طول رشته	size_t strlen(char *src);
strncat	اضافه کردن یک رشته در رشته دیگر	char *strcat(char *dest, char *src, size_t num);
strchr	پیدا کردن یک کاراکتر در رشته	char *strchr(char *str, char ch);
strstr	پیدا کردن یک رشته در رشته دیگر	char *strstr(char *str1, char *str2);
strcmp	مقایسه دو رشته	int strcmp(char *str1, char *str2);
gets	دریافت یک رشته از ورودی	char *gets(char *dest);
puts	نمایش یک رشته در خروجی	char *puts(const char *dest);

مثال. نحوه استفاده از تابع strlen و strcat.

```
#include <iostream.h>
#include <string.h>

int main()
{
   char firststring[40],secondstring[40],thirdstring[40];
   int size;

   cout << "Please enter a word \n";
   cin.getline(firststring,40);

   cout << "Please enter another word  \n";
   cin.getline(secondstring,40);
   size = strlen(firststring);

   strcat(firststring,secondstring);
   cout << "The length of the first string you entered is" << size << "\n";
   cout << "Both strings you entered are " << thirdstring<< "\n";
   return 0;
 }
```