

فصل دهم

«رشته‌ها و نحوه دستیابی به آنها»

دانشگاه صنعتی امیرکبیر
دکتر فخرناز قاسمی
ترم اول ۱۳۹۵

- ۷ حلقه‌ها و پرش‌ها for, While, ...
- ۸ توابع
- ۹ آرایه‌های یک بعدی و چند بعدی
- ۱۰ رشته‌ها و نحوه دستیابی به آنها
- ۱۱ کار با فایل‌ها
- ۱۲ اشاره‌گرها
- ۱۳ ساختارها و یونیون‌ها



اجتمالی

مقدمه

رشته کاراکتری: یک سلسله از کاراکترهای کنار هم در حافظه، پایان با کاراکتر NULL (۰)

ویژگی رشته‌های کاراکتری: آرایه‌ای از کاراکترها

۱- انتها: کاراکتر NULL  تعداد کل کاراکترها: طول رشته + ۱

۲- مقدارگذاری مستقیم رشته کاراکتری با لیترال رشته‌ای: `char str[] = "string";`

۳- چاپ رشته کاراکتری مثل یک متغیر معمولی: `cout << str;`

۴- دریافت رشته کاراکتری مثل یک متغیر معمولی: `cin >> str;`

آرایه `str` باید برای دریافت همه کاراکترهای وارد شده جا داشته باشد.

۵- توابع تعریف شده در سر فایل `<cstring>`

مثال ۱) رشته‌های کاراکتری

```
int main()
{ char s[] = "ABCD";
  for (int i = 0; i < 5; i++)
    cout << "s[" << i << "] = " << s[i] << "\n";
}
```

`s[0] = 'A'`
`s[1] = 'B'`
`s[2] = 'C'`
`s[3] = 'D'`
`s[4] = "`

S	
0	A
1	B
2	C
3	D
4	φ

`cin >> s;` →
 ↓

`getline(cin, s);`
`<string.h>`
`<cstring>`

فقط محدود به رشته‌ای که space ندارد. مثلاً `I am a student` را نمی‌توان گرفت.
 می‌توان تا زمانی که **Enter** زده نشده از آن استفاده کرد.

آرایه‌ای از رشته‌ها:

`char name[5][20];`

این آرایه در اصل پنج عضو دارد که هر عضو می‌تواند بیست کاراکتر داشته باشد. از طریق `name[0]` و `name[1]` و `name[2]` و `name[3]` و `name[4]` می‌توانیم به هر یک از رشته‌های کاراکتری در آرایه دسترسی داشته باشیم.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0																				
1																				
2																				
3																				
4																				

مثال ۲) برنامه‌ای بنویسید که یک رشته را گرفته و طول رشته را چاپ کند:

```
#include <iostream>
using namespace std;
int main()
{
    char a[64];
    int i,c;
    cout<<"Enter a strings:"<<endl;
    cin>>a;
    for(i=0; a[i]!='\0'; i++);
    cout<<"Size of string : "<<i<<endl;
    return 0;
}
```

```
#include <iostream>
using namespace std;
void size(char a[])
{
    int i;
    for(i=0; a[i]!='\0'; i++);
    cout<<"Size of string : "<<i<<endl;
}
int main()
{
    char a[64];
    cout<<"Enter a strings:"<<endl;
    cin>>a;
    size(a);
    return 0;
}
```

مثال ۳) برنامه‌ای بنویسید که دو مقدار برای رشته را گرفته و دومی را در اولی کپی کند

```
#include <iostream>
using namespace std;
int main()
{
    char a[64], b[64];
    cout<<"Enter 2 strings:"<<endl;
    cin>>a;
    cin>>b;
    int i=0;
    do
    {
        a[i]=b[i];
    }
    while(b[i++]!='\0');
    a[i]='\0';
    return 0;
}
```

توابع استاندارد رشته های کاراکتری

سرفایل **<cstring>**: کتابخانه رشته‌های کاراکتری شامل خانواده توابعی بسیار مفید برای کار با رشته‌های کاراکتری.

strlen (s) ; طول رشته را برمی‌گرداند
strcpy (s1, s2) ; s2 را داخل s1 کپی می‌کند (مقتوای s1 از بین می‌رود)
strcat (s1, s2) ; s2 را به انتهای s1 می‌پسباند
strcmp (s1 , s2) ; s1 , s2 را مقایسه کرده و عددی برمی‌گرداند

$0 \Rightarrow$ برابری

$+\Rightarrow$ اول s2 بعد s1

$-\Rightarrow$ اول s1 بعد s2

```
s1="Mohammad";
strcpy (s2 , "Ali" );
strcat (s1 , s2 );
cout<<s1;
```

ترتیب را در حروف الفبا داریم

```
strcmp ("Ali" , "Ali") → 0
strcmp ("Ali" , "ali") → -1
strcmp ("Ali" , "Babak") → -1
strcmp ("Babak", "Ali") → +1
```



توابع تعریف شده در سرفایل <cstring>

شرح	نام تابع
Char* strcat (char* s1, const char* s2);	Strcat () S2 را به s1 الحاق می کند s1 را برمی گرداند.
Char* strchr (const char* s, int c);	Strchr () یک اشاره گر به اولین وقوع c در رشته s برمی گرداند. اگر c در s نباشد، NULL را برمی گرداند.
Int strcmp (const char* s1, const char* s2);	Strcmp () S1 را با زیررشته S2 مقایسه می کند. اگر S1 به ترتیب الفبایی، کوچکتر یا مساوی یا بزرگتر از S2 باشد مقدار منفی یا صفر یا مقدار مثبت را برمی گرداند.
char* strcpy(char* s1, const char* s2);	Strcpy () S2 را در S1 کپی می کند S1 را برمی گرداند.
Size _t strcspn (char* s1, const char* s2);	Strcspn () طول بزرگترین زیررشته ای از s1 را برمی گرداند که با s[0] شروع شده و شامل هیچ یک از کاراکترهای موجود در s2 نیست.
Size _t strlen (const char* s);	strlen () طول s را برمی گرداند که تعداد کاراکترهایی است که با s[0] شروع می شود و با اولین کاراکتر null خاتمه می یابد.

Char* strncat (char* s1, const char* s2, size_t n);	n کاراکتر اول s2 را به s1 الصاق می کند اگر n > strlen (s2) باشد، آنگاه strncat (s1,s2,n) تأثیری مشابه strcat (s1,s2) دارد.	Strncat ()
Int strncmp (const char* s1, const char* s2 size_t n)	n کاراکتر اول s1 را با n کاراکتر اول s2 مقایسه می کند و اگر زیررشته اول به ترتیب الفبایی از زیررشته دوم بزرگتر، مساوی یا کوچکتر باشد مقدار مثبت، صفر یا مقدار منفی را برمی گرداند. اگر n > strlen (s2) باشد، آنگاه strncmp (s1,s2,n) تأثیری مشابه با strcmp (s1, s2) خواهد داشت.	Strncmp ()
Char* strncpy (char* s1, const char* s2, size_t n);	n کاراکتر اول s1 را با n کاراکتر اول s2 جایگزین می کند و s1 را برمی گرداند. اگر n < strlen (s1) باشد، طول s1 تغییر نمی کند. اگر n > strlen (s2) باشد آنگاه strncpy (s1,s2,n) تأثیری مشابه strcpy (s1,s2) خواهد داشت.	Strncpy ()
Char* strpbrk (const char* s1, const char* s2);	محل اولین رخداد هریک از کاراکترهای s2 را در s1 برمی گرداند. اگر هیچ یک از کاراکترهای s2 در s1 یافت نشد، NULL را برمی گرداند.	Strpbrk ()
Char* strrchr (const char* s, int c);	آخرین محل قرار گرفتن کاراکتر c در رشته s را برمی گرداند. اگر c در s نباشد، NULL را برمی گرداند.	Strrchr ()
Size_t strepn (char* s1, const char* s2);	طول بزرگترین زیررشته ای از s1 را برمی گرداند که از s[0] شروع شده و فقط شامل کاراکترهای موجود در s2 است.	Strspn ()

Char* strstr (const char* s1, const char* s2);	آدرس اولین محل وقوع زیررشته s2 در رشته s1 را برمی گرداند اگر s2 در s1 نباشد، NULL را برمی گرداند.	Strstr ()
Char* strtok (char* s1, char* s2);	رشته s1 را با استفاده از کاراکترهای موجود در رشته s2 نشانه گذاری می کند پس از فراخوانی آغازین strtok (s1, s2) هر فراخوانی موفقیت آمیز، strtok (s1, s2) اشاره گری به نشانه یافت شده بعدی در s1 برمی گرداند. این فراخوانی ها رشته s1 را تغییر می دهد و هر کاراکتر نشانه را با کاراکتر NULL جایگزین می کند.	Strtok ()

مثال (۴) تابع strlen()

```
#include <cstring>
#include <iostream>
using namespace std;
int main()
{
    char s[] = "ABCDEFGH";
    cout << "strlen(" << s << ") = " << strlen(s) << endl;
    cout << "strlen(\"\\\") = " << strlen("") << endl;
    char buffer[80];
    cout << "Enter string: "; cin >> buffer;
    cout << "strlen(" << buffer << ") = " << strlen(buffer) << endl;
}
```

رشته‌های کاراکتری

رشته‌های کاراکتری را نمی‌توانیم با استفاده از عملگر جایگزینی (=) درون یکدیگر کپی کنیم (پرا؟) اما دو تابع وجود دارد که عمل جایگزینی را شبیه سازی می‌نمایند. تابع strcpy(s1, s2) باعث می‌شود که رشته کاراکتری S2 درون رشته کاراکتری S1 کپی شود. همچنین تابع strncpy(s1, s2, n) باعث می‌شود که n کاراکتر اول از رشته s2 روی n کاراکتر اول رشته s1 کپی شود. هر دو تابع فوق s1 را برمی‌گردانند و S2 بدون تغییر خواهد ماند.


```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
```

```
    char s1[] = "ABCDEFGH";
    char s2[] = "XYZ";
    cout << "Before strcpy(s1,s2):\n";
    cout << "\ts1 = [" << s1 << "], length = " << strlen(s1)<< endl;
    cout << "\ts2 = [" << s2 << "], length = " << strlen(s2)<< endl;
    strcpy(s1,s2);
    cout << "After strcpy(s1,s2):\n";
    cout << "\ts1 = [" << s1 << "], length = " << strlen(s1)<< endl;
    cout << "\ts2 = [" << s2 << "], length = " << strlen(s2)<< endl;
}
```

مثال (۵) تابع strcpy()

Before strcpy(s1,s2):

s1 = [ABCDEFGH], length = 7

Before strcpy(s1,s2):

s1 = [ABCDEFGH], length = 7

s2 = [XYZ], length = 3

After strcpy(s1,s2):

s1 = [XYZ], length = 3

s2 = [XYZ], length = 3

فراخوانی strncpy(s1,s2,2) باعث می شود که دو کاراکتر اول رشته S2، روی دو کاراکتر اول رشته S1 کپی شود. طول S2 تاثیری بر طول S1 ندارد و اندازه S1 تغییر نمی کند.

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
```

```
    char s1[] = "ABCDEFGH";
    char s2[] = "XYZ";
    cout << "Before strncpy(s1,s2,2):\n";
    cout << "\ts1 = [" << s1 << "], length = " << strlen(s1)<< endl;
    cout << "\ts2 = [" << s2 << "], length = " << strlen(s2)<< endl;
    strncpy(s1,s2,2);
    cout << "After strncpy(s1,s2,2):\n";
    cout << "\ts1 = [" << s1 << "], length = " << strlen(s1)<< endl;
    cout << "\ts2 = [" << s2 << "], length = " << strlen(s2)<< endl;
}
```

مثال (۶) تابع strncpy()

Before strncpy(s1,s2,2):

s1 = [ABCDEFGH], length = 7

s2 = [XYZ], length = 3

After strncpy(s1,s2,2):

s1 = [XYCDEFGH], length = 7

s2 = [XYZ], length = 3

در فراخوانی `strncpy(s1,s2,n)` اگر `strlen(s1)>n` باشد، آنگاه `n` کاراکتر اول `s2` روی `n` کاراکتر اول `s1` کپی می شود و اگر `strlen(s1)<=n` باشد، آنگاه تاثیر این تابع با تابع `strcpy()` یکی خواهد بود.

دو تابع `strcat()` و `strncat()` همانند توابع `strcpy()` و `strncpy()` رفتار می کنند با این تفاوت که این توابع، کاراکترهای رشته `s2` را به انتهای رشته `s1` الحاق می کنند. عبارت "cat" از کلمه "catenate" به معنای «الحاق نمودن» گرفته شده. البته دقت کنید که توابع مذکور `s1` و `s2` را به یک رشته واحد تبدیل نمی کنند بلکه یک کپی از کاراکترهای `s2` را به انتهای `s1` پیوند می زنند.

مثال (۷) تابع الحاق رشته `strcat()`

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
```

```
    char s1[] = "ABCDEFGH";
    char s2[] = "XYZ";
    cout << "Before strcat(s1,s2):\n";
    cout << "\ts1 = [" << s1 << "], length = " << strlen(s1)<< endl;
    cout << "\ts2 = [" << s2 << "], length = " << strlen(s2)<< endl;
    strcat(s1,s2);
    cout << "After strcat(s1,s2):\n";
    cout << "\ts1 = [" << s1 << "], length = " << strlen(s1)<< endl;
    cout << "\ts2 = [" << s2 << "], length = " << strlen(s2)<< endl;
}
```

Before `strcat(s1,s2):`

`s1 = [ABCDEFGH], length = 7`

`s2 = [XYZ], length = 3`

After `strcat(s1,s2):`

`s1 = [ABCDEFGHXYZ], length = 10`

`s2 = [XYZ], length = 3`

مثال ۱) تابع الصاق رشته strncat()

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{ // test-driver for the strncat() function:
  char s1[] = "ABCDEFGH";
  char s2[] = "XYZ";
  cout << "Before strncat(s1,s2,2):\n";
  cout << "\ts1 = [" << s1 << "], length = " << strlen(s1)<< endl;
  cout << "\ts2 = [" << s2 << "], length = " << strlen(s2)<< endl;
  strncat(s1,s2,2);
  cout << "After strncat(s1,s2,2):\n";
  cout << "\ts1 = [" << s1 << "], length = " << strlen(s1)<< endl;
  cout << "\ts2 = [" << s2 << "], length = " << strlen(s2)<< endl;
}
```

Before strncat(s1,s2):

s1 = [ABCDEFGH], length = 7

s2 = [XYZ], length = 3

After strncat(s1,s2):

s1 = [ABCDEFGHXY], length = 9

s2 = [XYZ], length = 3

توابع کاراکتری C استاندارد

نام تابع	شرح
isalnum ()	Int isalnum (int c); اگر C کاراکتر الفبایی یا عددی باشد مقدار غیر صفر و گرنه صفر را برمیگرداند
isalpha ()	Int isalpha (int c); اگر C کاراکتر الفبایی باشد مقدار غیر صفر و در غیر آن، صفر را برمی گرداند.
isctrl ()	Int isctrl (int c); اگر C کاراکتر کنترلی باشد مقدار غیر صفر و در غیر آن، صفر را برمی گرداند.
isdigit ()	Int isdigit (int c); اگر C کاراکتر عددی باشد، مقدار غیر صفر و در غیر آن، صفر را برمی گرداند.
isgraph ()	Int isgraph (int c); اگر C کاراکتر چاپی و غیر خالی باشد مقدار غیر صفر و گرنه صفر را برمی گرداند.
islower ()	Int islower (int c); اگر C حرف کوچک باشد مقدار غیر صفر و در غیر آن، صفر را برمی گرداند.
isprint()	Int isprint (int c); اگر C کاراکتر قابل چاپ باشد مقدار غیر صفر و در غیر آن، صفر را برمی گرداند.

ispunct ()	Int ispunct (int c);	اگر C کاراکتر چاپی به غیر از حرف و اعداد و فضای خالی باشد، مقدار غیرصفر برمی گرداند وگرنه مقدار صفر را برمی گرداند.
isspace ()	Int isspace (int c);	اگر C کاراکتر فضای سفید شامل فضای خالی " " و عبور فرم 'f' و خط جدید 'n' و بازگشت نورد 'r' و پرش افقی 't' و پرش عمودی 'v' باشد، مقدار غیرصفر را برمی گرداند وگرنه صفر را برمی گرداند.
isupper ()	Int isupper (int c);	اگر C حرف بزرگ باشد، مقدار غیرصفر برمی گرداند وگرنه صفر را برمی گرداند
isxdigit ()	Int isxdigit (int c);	اگر C یکی از ده کاراکتر عددی یا یکی از دوازده حرف عدد شانزدهی شامل 'a' و 'b' و 'c' و 'd' و 'e' و 'f' و 'A' و 'B' و 'C' و 'D' و 'E' و 'F' باشد، مقدار غیرصفر برمی گرداند وگرنه مقدار صفر را برمی گرداند.
tolower ()	Int tolower (int c);	اگر C حرف بزرگ باشد، کاراکتر کوچک معادل آن را برمی گرداند وگرنه خود C را برمی گرداند.
toupper ()	Int toupper (int c);	اگر C حرف کوچک باشد، کاراکتر بزرگ معادل آن را برمی گرداند وگرنه خود C را برمی گرداند.

نوع string در C++ استاندارد

مثل رشته های کاراکتری برای ذخیره کردن مجموعه ای از کاراکترها به کار می روند.

سرفایل <string>

طریقه اعلان و مقداردهی:

```
string s1; // s1 contains 0 characters
string s2 = "AUT University"; // s2 contains 14 characters
string s3(60, '*'); // s3 contains 60 asterisks
string s4 = s3; // s4 contains 60 asterisks
string s5(s2, 4, 2); // s5 is the 2-character string "Un"
```

String ها را بر خلاف رشته های کاراکتری می توانیم به یکدیگر تفهیم و آنها را از روی یک شیء string موجود مقداردهی کنیم.

سازنده زیر رشته سه قسمت دارد:

۱- رشته والد که زیر رشته از درون آن استخراج می شود (در این جا S2 است)

۲- کاراکتر آغازین زیر رشته (در این جا s[4] است)

۳- طول زیر رشته (در این جا ۲ است).

ورودی قالب بندی شده با stringها مثل رشته های کاراکتری معمولی رفتار می کند. یعنی هنگام وارد کردن کاراکترهای دریافتی، کاراکترهای فضای سفید را نادیده گرفته و حذف می کند و همین که بعد از کلمه جاری به یک کاراکتر فضای سفید برسد، دریافت کاراکترها را فاصله می دهد.

getline()

1.

```
string s = "ABCDEFGH";
```

```
getline(cin, s); // reads the entire line of characters into s
```

همچنین درون stringها می توانیم از عملگر اندیس مثل رشته های کاراکتری استفاده کنیم:

2.

```
char c = s[2]; // assigns 'C' to c
```

```
s[4] = '*'; // changes s to "ABCD*FG"
```

* دقت کنید که اینجا هم ایندکس از صفر شروع می شود.

تبدیل اشیای نوع string به نوع رشته کاراکتری

۳. یافتن تعداد کاراکترهای موجود در یک شی

```
cout << s.length() << endl; // prints 7 for the string s = "ABCD*FG"
```

۴. مقایسه stringها با استفاده از عملگرهای رابطه‌ای

```
if (s2 < s5) cout << "s2 lexicographically precedes s5\n";
while (s4 == s3) // ...
```

۵. پیوند محتویات stringها به یکدیگر با استفاده از عملگرهای + و +=

```
string s6 = s + "HIJK"; // changes s6 to "ABCD*FGHIJK"
s2 += s5; // changes s2 to "PNU UniversityUn"
```

۷. استخراج یک زیررشته از درون یک string:

```
s4 = s6.substr(5,3); // changes s4 to "FGH"
```

۸. حذف یا رونویسی بخشی از محتویات درون یک string:

```
s6.erase(4, 2); // changes s6 to "ABCDGHIJK"
s6.replace(5, 2, "xyz"); // changes s6 to "ABCDGxyzJK"
```

erase(): پارامتر اول، نقطه شروع حذف و پارامتر دوم، تعداد کاراکترهایی که باید حذف شوند.

replace(): پارامتر اول، نقطه شروع رونویسی، پارامتر دوم تعداد کاراکترهایی که باید حذف شوند و پارامتر سوم زیررشته‌ای است که باید به جای کاراکترهای حذف شده قرار بگیرد.

۹. ایندکس اولین وقوع یک زیررشته مفروض در string فعلی:

```
string s7 = "The SOFTWARE MOVEMENT bases";
cout << s7.find("EM") << endl; // prints 16
cout << s7.find("EO") << endl;
// prints 27, the length of the string
```

اگر تابع find() زیررشته موردنظر را پیدا نکند، صفر (در نسخه های قدیمی تر طول رشته تحت جستجو) را برمی گرداند.



0 0
'0' 48

مثال ۹) یک رشته از اعداد صحیح

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    bool correct=true ;
    string s ;
    int n=0, i, L;
    cout<<"Enter a string of numbers :";
    cin>> s;
    L=s.length ( );
    for (i=0 ; i<L && correct; i++)
    {
        if ((s[i]>='0') && (s[i]<='9'))
            n=10*n+(s[i]-'0');
        else
            correct = false ;
    }
    if (correct)
        cout << n << endl;
    else
        cout<<"your string is not in correct format" << endl;
    return 0 ;
}
```



مثال ۱۰) برنامه‌ای بنویسید که یک رشته حداکثر ۱۰۰ حرفی از ورودی گرفته و بزرگی و کوچکی حروف آن را برعکس کند.

```
#include <string>
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{ char s[101];
```

```
int i, L;
```

```
cout<< "Enter your string: ";
```

```
cin.getline (s,100) ;
```

```
L = strlen(s) ;
```

```
for (i = 0 ; i<L ; i++)
```

```
{ if ((s[i]>='A') && (s[i]<= 'Z'))
```

```
s[i]+='a'-'A' ;
```

```
else
```

```
if ((s[i]>='a') && (s[i]<= 'z'))
```

```
s[i]+='A'-'a' ;
```

```
}
```

```
cout<<s << endl;
```

```
return 0 ;
```

```
}
```

29/48

دانشگاه صنعتی امیرکبیر - دکتر قاسمی

12/22/2016

مثال ۱۲) برنامه‌ای بنویسید که با گرفتن یک رشته تعداد کلمات آن را بشمرد. (کلمات با space و tab از هم جدا شده‌اند)

```
#include <string>
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
string s;
```

```
int i=0, k=0, L ;
```

```
cout<<"Enter your string:";
```

```
getline(cin, s) ;
```

```
L=s.length ( ) ;
```

```
while(i<L)
```

```
{
```

```
while((s[i]==32)|| (s[i]==9))
```

```
i++;
```

```
if (i<L)
```

```
k++ ;
```

```
while ((s[i]!=32) && (s[i]!=9) &&(i<L))
```

```
i++ ;
```

```
}
```

```
cout<<" Number of words in \" << s << "\" is :\" << k << endl;
```

```
return 0 ;
```

```
}
```

32/48

دانشگاه صنعتی امیرکبیر - دکتر قاسمی

12/22/2016

رشته‌های کاراکتری در C++ استاندارد

تبادل داده‌ها:

وقتی می‌خواهیم داده‌هایی را وارد کنیم، این داده‌ها را در قالب مجموعه‌ای از کاراکترها تایپ می‌کنیم. همچنین وقتی می‌خواهیم نتایجی را به خارج از برنامه بفرستیم، این نتایج در قالب مجموعه‌ای از کاراکترها نمایش داده می‌شوند.

تفسیر این کاراکترها برای برنامه



جریان ها: تبدیل داده‌ها به کاراکتر و
کاراکترها به داده‌هایی از یک نوع بنیادی



ورودی‌ها و خروجی‌ها: کلاس جریان به نام stream

- شیء **istream**: جریانی است که داده‌های مورد نیاز را از کاراکترهای وارد شده از صفحه کلید، فراهم می‌کند.
- شیء **ostream**: جریانی است که داده‌های حاصل را به کاراکترهای خروجی قابل نمایش روی صفحه نمایشگر تبدیل می‌نماید.
- شیء **ifstream**: جریانی است که داده‌های مورد نیاز را از داده‌های داخل یک فایل، فراهم می‌کند.
- شیء **ofstream**: جریانی است که داده‌های حاصل را درون یک فایل ذخیره می‌نماید.



کلاس istream

نمونه رفتار با کاراکترهای ورودی را تعریف می‌نماید.
«عملگر برون‌کشی >>» یا عملگر ورودی، دارای دو عملوند:

- ۱- شیء `istream` که مشخص می‌کند کاراکترها از کجا باید بیرون کشیده شوند و
 - ۲- شیئی که مشخص می‌کند مقدار برون‌کشی شده باید از چه نوعی باشد و کجا باید ذخیره شود.
- به این پردازش که از کاراکترهای خام ورودی مقادیری با نوع مشخص تولید می‌کند، **قالب بندی (Formatting)** می‌گویند.

عملگر برون‌کشی >> : ورودی را **قالب بندی** می‌کند.

```
int n;  
cin >> n;
```

“ 46”

شیء جریان `cin` کاراکترها را یکی یکی پویش می‌کند. اگر اولین کاراکتری که به آن وارد می‌شود، کاراکتر فضای خالی یا هر کاراکتر فضای سفید دیگر (مثل `tab` یا فاصله) باشد، آن را نادیده گرفته و از جریان ورودی حذف می‌کند. این کار ادامه می‌یابد تا این که به یک کاراکتر غیرفاصله‌ای برخورد کند.

- ۱- مقداری برای `n` فراهم می‌کند،
- ۲- حاصل دیگری به شکل یک داده منطقی برحسب این که عمل برون‌کشی موفقیت آمیز بوده یا خیر، مقدار `true` یا `false` دارد.

مثال ۱۳) استفاده از عملگر بیرون کشی برای کنترل کردن یک حلقه

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    while (cin >> n)
        cout << "n = " << n << endl;
}
```

46

n = 46

22 44 66 88

n = 22

n = 44

n = 66

n = 88

33, 55, 77, 99

n = 33

ورودی / خروجی رشته‌های کاراکتری (چند تابع عفو cin و cout) ورودی قالب بندی نشده

سرفایل <iostream>

cin: شیء فرآیند ورودی شامل:

cin.getline(), cin.get(), cin.ignore(), cin.putback(), cin.peek()

تابع cin.get() برای دریافت یک کاراکتر تکی

تابع cin.getline() برای دریافت یک رشته کاراکتری

فراخوانی cin.getline(str,n); باعث می‌شود که n کاراکتر به درون str خوانده شود و مابقی کاراکترهای وارد شده نادیده گرفته می‌شوند.

cout: شیء فرآیند خروجی شامل cout.put() است.

مثال ۱۴) تابع cin.get() و cout.put()

```
#include <iostream>
using namespace std;
int main()
```

```
{
```

```
    char ch;
```

```
    int count = 0;
```

```
    while (cin.get(ch))
```

```
        if (ch == 'e') ++count;
```

```
    cout << count << " e's were counted.\n";
```

```
    char pre = '\0';
```

```
    while (cin.get(ch))
```

```
    {
```

```
        if (pre == ' ' || pre == '\n')
```

```
            cout.put(char(toupper(ch)));
```

```
        else cout.put(ch);
```

```
        pre = ch;
```

```
    }
```

```
}
```

cin.get(ch): برای خواندن یک کاراکتر از ورودی

کاراکتر بعدی از ورودی cin خوانده شده و به داخل متغیر ch کپی می شود.

cout.put(): معکوس تابع ورودی get()

برای نوشتن یک کاراکتر در خروجی

۷

۷) حلقه ها و پرش ها for, While, ...

۸

۸) توابع

۹

۹) آرایه های یک بعدی و چند بعدی

۱۰

۱۰) رشته ها و نحوه دستیابی به آنها

۱۱

۱۱) کار با فایل ها

۱۲

۱۲) اشاره گر ها

۱۳

۱۳) ساختارها و یونیون ها

اعتمادی

فایل ها

فایل ها: قدرت نگهداری اطلاعات مجیم
 امکان ارتقاء ذخیره، بازیابی و کارایی برنامه ها
 پردازش **فایل** در C++ بسیار شبیه ترانکشن های معمولی ورودی و خروجی
 است زیرا اینها همه از اشیای جریان مشابهی بهره می برند.



جریان **fstream** برای ترانکشن برنامه با فایل ها به کار می رود با دو زیرشافه:
ifstream: برای خواندن اطلاعات از یک فایل
ofstream: برای نوشتن اطلاعات درون یک فایل

```
#include <fstream>
```

```
ifstream readfile("INPUT.TXT");
```

```
ofstream writefile("OUTPUT.TXT");
```

readfile: داده ها را از فایلی به نام INPUT.TXT می خواند

Writefile: اطلاعاتی را در فایلی به نام OUTPUT.TXT می نویسد.

می توان با استفاده از عملگر >> داده ها را از درون **readfile** خواند و با
 عملگر << اطلاعات را درون **writefile** نوشت.



مثال ۱۵) برنامه‌ای بنویسید که یک دفتر تلفن را ذخیره و در فایلی به نام PHONE.TXT ذخیره کند. برای پایان دادن به ورودی عدد ۰ تایپ گردد.

```
#include <fstream>
#include <iostream>
#include <string>
using namespace std;
int main()
{
    long number=0;
    string name;
    ofstream phonefile("PHONE.TXT");
    cout << "Enter a number for each name. (0 for quit): ";
    for ( ; ; )
    {
        cout << "Number: ";
        cin >> number;
        if (number == 0) break;
        phonefile << number << ' ';
        cout << "Name: ";
        cin >> name;
        phonefile << name << ' ';
        cout << endl;
    }
}
```

مثال ۱۶) جستجوی یک شماره در دفتر تلفن

```
#include <fstream>
#include <iostream>
#include <string>
using namespace std;
int main()
{
    ifstream phonefile("PHONE.TXT");
    long number;
    string name, searchname;
    bool found=false;
    cout << "Enter a name for finding it's phone number: ";
    cin >> searchname;
    cout << endl;
    while (phonefile >> number)
    {
        phonefile >> name;
        if (searchname == name)
        {
            cout << name << ' ' << number << endl;
            found = true;
        }
    }
    if (!found)
        cout << searchname << " is not in this phonebook." << endl;
}
```

خواندن اطلاعات از درون فایل به صورت کاراکتر به کاراکتر: `get()`
 نوشتن اطلاعات به شکل کاراکتر به کاراکتر درون یک فایل: `put()`
فایل های متنی: فایل هایی که اطلاعات درون آنها به صورت متنی ذخیره می شود و به وسیله برنامه های واژه پرداز می توان اطلاعات درون این فایل ها را دیده و ویرایش کرد.
فایل های دودویی (باینری): اطلاعات درون این فایل ها به شکل کدهای اسکی ذخیره می شوند و معمولا به سادگی نمی توان فهمید که چه اطلاعاتی درون آنها است.

مثال ۱۷) برنامه ای بنویسید که از فایل `input.txt`

اولین عدد را به عنوان `n` در نظر گرفته و `n` عدد از فایل بخواند

```
#include <fstream>
#include <iostream>
#include <conio.h>
using namespace std;
int main()
```

```
{
    int n, a, i;
    ifstream in;
    in.open ("input.txt");
    if (in. fail() )
        cout<< "Error Opening input.txt";
    else
    {
        in>> n;
        cout << "This Program will read " << n << " numbers from the file:" << endl;
        for (i=0 ; i<n ; i++)
        {
            in>> a;
            cout<< a << endl ;
        }
        in.close( ) ;
    }
    getch ( ) ;
    return 0 ;
}
```

تعریف فایل از نوع ورودی

فایل `input.txt` باز می شود

در صورتی که فایل همان جا نباشد "C:\\new folder 1\\input.txt"

نتوانست باز کند

اولین عدد فایل وارد `n` می شود

فایل بسته resource آن آزاد می شود

مثال ۱۸) در فایل input.txt تعدادی عدد اعشاری

موجود است. برنامه‌ای بنویسید که معدل این اعداد را

محاسبه کند.

```
#include <fstream>
#include <iostream>
#include <conio.h>
using namespace std;
```

```
int main()
```

```
{
    int n = 0 ;
```

```
    double s = 0 , a ;
```

```
    ifstream in ;
```

```
    in.open("input.txt") ;
```

```
    if ( in.fail() )
```

```
        cout<<"Error opening input.txt" ;
```

```
    else
```

```
    {
```

```
        while (!in.eof() )
```

```
        {
            in >> a;
```

```
            if (in.good() )
```

```
            {
                s+= a ;
```

```
                n++ ;
```

```
            }
```

```
        }
        in.close() ;
```

```
        if (n>0)
```

```
        {
            s /=n ;
```

```
            cout<<"Average="<<s << endl;
```

```
        }
```

```
    }
    else
```

```
        cout<<"there is no number";
```

```

}
```

```
getch() ;
```

```
return 0 ;
```

```
}
```

47/48

دانشگاه صنعتی امیرکبیر - دکتر قاسمی

12/22/2016

پرسش و پاسخ ؟

48/48

12/22/2016