In The Name of Allah

برنامه نویسی به زبان C++

فهرست مطالب

- مباحث تکمیلی در رشته ها
  - ورودي/خروجي رشته‌هاي کاراکتري
  - چند تابع عضو cin و cout
  - آرايه‌اي از رشته‌ها
  - فايلها

ورودي/خروجي رشته‌هاي كاراكتري

- در C++به چند روش مي‌توان رشته‌هاي كاراكتري را دريافت كرده يا نمايش داد.
- يک راه استفاده از عملگرهاي كلاس stringاست كه در بخش‌هاي بعدي به آن خواهيم پرداخت.
- روش ديگر، استفاده از توابع كمكي است كه آن را در ادامه شرح مي‌دهيم.

ورودي/خروجي رشته‌هاي كاراكتري

- روش سادهٔ دريافت و نمايش رشته‌هاي كاراكتري
- در برنامهٔ زير يک رشتهٔ كاراكتري به طول 20 كاراكتر اعلان شده و كلماتي كه از ورودي خوانده مي‌شود در آن رشته قرار مي‌گيرد:

```
int main()
{  char word[20];
   do
   {  cin >> word;
      if (*word) cout << "\t\"" << word << "\"\n";
   } while (*word);
}
```

چند تابع عضو cin و cout

- به cin شيء فرآيند ورودي مي‌گويند. اين شي شامل توابع زير است:

cin.getline()

cin.get()

cin.ignore()

cin.putback()

- همهٔ اين توابع شامل پيشوند cin هستند زيرا آن‌ها عضوي از cin مي‌باشند.
به cout شيء فرآيند خروجي مي‌گويند. اين شي نيز شامل تابع cout.put() است. نحوهٔ كاربرد هر يک از اين توابع عضو را در ادامه خواهيم ديد.

تابع cin.getline() با دو پارامتر

- این برنامه ورودی را خط به خط به خروجی می‌فرستد:

```
int main()
{  char line[80];
   do
   {  cin.getline(line,80);
      if (*line) cout << "\t[" << line << "]\n";
   } while (*line);
}
```

تابع cin.getline() با سه پارامتر

- برنامه زير، متن ورودي را جمله به جمله تفكيک مي‌نمايد:

```
int main()
{  char clause[20];
   do
   {  cin.getline(clause, 20, ',');
      if (*clause) cout << "\t[" << clause << "]\n";
   } while (*clause);
}
```

تابع cin.get()

- این تابع حرف به حرف داده ها را از ورودی می خواند
- این برنامه تعداد حرف ‘e’ در جريان ورودي را شمارش مي‌كند.

```
int main()
{  char ch;
   int count = 0;
   while (cin.get(ch))
      if (ch = = 'e') ++count;
   cout << count << " e's were counted.\n";
}
```

تابع cin.get() با دوپارامتر ورودی

- خواندن رشته ها با فواصل خالی

```
int main()
{   const int Max=20;
char clause[Max];
    cin.get(clause, Max, );
    cout << clause ;

}
```

cin.ignore() وcin.putback() توابع

```
int nextInt();
int main()
{   int m = nextInt(), n = nextInt();
  cin.ignore(80,'\n');          // ignore rest of input line
   cout << m << " + " << n << " = " << m+n << endl;
}
int nextInt()
{  char ch;
   int n;
   while (cin.get(ch))
      if (ch >= '0' && ch <= '9')  // next character is a digit
      {  cin.putback(ch);          // put it back so it can be
         cin >> n;                 // read as a complite int
         break;
      }
   return n;
}
```

آرايه‌اي از رشته‌ها

- به خاطر داريد كه گفتيم يک آرايهٔ دوبعدي در حقيقت آرايه‌اي يک بعدي است كه هر كدام از اعضاي آن يک آرايهٔ يک بعدي ديگر است. مثلا در آرايهٔ دو بعدي كه به شكل مقابل اعلان شده باشد:

```
char name[5][20];
```

از طريق name[0] وname[1] وname[2] وname[3] وname[4] مي‌توانيم به هر يک از رشته‌هاي كاراكتري در آرايهٔ بالا دسترسي داشته باشيم. يعني آرايهٔ nameگرچه به صورت يک آرايهٔ دوبعدي اعلان شده ليكن به صورت يک آرايهٔ يک بعدي با آن رفتار مي‌شود.

آرايه‌اي از رشته‌هاي كاراكتري

برنامهٔ زير چند رشتهٔ كاراكتري را از ورودي مي‌خواند و آن‌ها را در يک آرايه ذخيره كرده و سپس مقادير آن آرايه را چاپ مي‌كند:

```
int main()
{  char name[5][20];
   int count=0;
   cout << "Enter at most 4 names with at most 19 characters:\n";
   while (cin.getline(name[count++], 20)) ;
   --count;
   cout << "The names are:\n";
   for (int i=0; i<count; i++)
      cout << "\t" << i << ". [" << name[i] << "]" << endl;
}
```

فايل‌ها

- يكي از مزيت‌هاي رايانه، قدرت نگهداري اطلاعات حجيم است. فايل‌ها اين قدرت را به رايانه مي‌دهند.
- پردازش فايل در C++بسيار شبيه تراكنش‌هاي معمولي ورودي و خروجي است زيرا اين‌ها همه از اشياي جريان مشابهي بهره مي‌برند. جريان fstreamبراي تراكنش برنامه با فايل‌ها به كار مي‌رود. fstream نيز به دو زيرشاخهٔ ifstreamو ofstreamتقسيم مي‌شود.
- جريان ifstreamبراي خواندن اطلاعات از يک فايل به كار مي‌رود و جريان ofstreamبراي نوشتن اطلاعات درون يک فايل استفاده مي‌شود.

فایل‌ها

- **این جریان‌ها در سرفایل** <fstream> **تعریف شده‌اند.**
- مي‌توانيد عناصري از نوع جريان فايل به شكل زير تعريف كنيد:

```
ifstream readfile("INPUT.TXT");
ofstream writefile("OUTPUT.TXT");
```

- اکنون مي‌توان با استفاده از عملگر >> داده‌ها را به درون readfileخواند و با عملگر << اطلاعات را درون writefileنوشت. به مثال زیر توجه کنید.

خواندن و نوشتن فایل

- برای ساخت یک شی از کلاس ifstream (برای خواندن اطلاعات) به شکل رو به رو عمل می کنیم:

```
ifstream myObjectName( nameOfFile, flag);
```

- مثال

```
ifstream fp("my file.txt", flag);
```

flag: نوع فایل و طریقه خواندن را مشخص می کند.

- مثال

```
ifstream myObjectName( "my file.txt", ios::in | ios::binary)
```

خواندن و نوشتن فایل

- برای کار کردن با فایل ها باید هدرشان را به برنامه بیافزاییم.

```
#include <fstream> //file header
```

- بتدا باید یک فایل را ایجاد کنیم. در کد زیر یک شی از جریان خروجی فایل باز کنیم:

```
ofstream outputFile("output.dat", ios::out);
```

- همیشه بعد از ساخت یک شی از کلاس ifstreamو ofstreamباید چک کنیم و ببینیم که فایل درست باز شده است یا نه:

```
if ( !outputFile)
{
cerr << "some thing wrong during opening file!" << endl;
exit(1);
}
```

خواندن و نوشتن فایل

- فرض کنید متغییر رشته ای name و متغییر اعشاری scoreرا داریم
- برای خواندن آنها از یک فایل

```
inputFile >> name >> score;
```

- برای قرار دادن آن در یک فایل

```
outputFile << name << " " << score << endl;
```