عنوان مقاله:

آشنایی با مفاهیم ورودی و خروجی برای مسابقات ACM به زبان های C++ و Java

تهیه کننده:

محمدرضا رحمانی

گروه ACM دانشگاه آزاد اسلامی واحد پرند

بهار ۹۲

به نام خدا

مقدمه

حل اغلب سوالات در مسابقات برنامه نویسی نیازمند آن است که ابتدا اطلاعاتی را از ورودی (کیبورد/کنسول/stdin) خوانده و پس از اجرای الگوریتم روی آن ها جواب را در خروجی (کنسول/stdout) نمایش دهید. اگر خروجی برنامه ما با خروجی مطلوب سوال، **حتی فقط در یک ' ' (Space)** ، اختلاف داشته باشد برنامه شما پذیرفته نشده و غلط به حساب خواهد آمد و پنالتی خواهید خورد؛ **حتی اگر الگوریتم شما صحیح باشد!** از این رو و با توجه به اهمیت این موضوع لازم است که ویژگی ها و خصوصیات ورودی و خروجی را به طور دقیق بشناسیم که این مقاله به همین منظور نوشته شده است.

C++ - cin/cout

برای کار کردن با ورودی و خروجی در زبان C++ توصیه می شود که در آغاز از cin و cout استفاده کنید. برای استفاده از cin و cout کد زیر را به ابتدای کد خود اضافه کنید:

```
#include <iostream>   // cin, cout برای استفاده از
#include <string>     // string برای استفاده از کلاس

using namespace std;
```

حال قادر خواهید بود تا از cin و cout استفاده کنید:

```
int number;                                 // number تعریف عدد صحیح
string word;                                // word تعریف رشته
cin >> number >> word;                      // word و رشته number خواندن عدد
cout << number << " " << word << endl; // word و رشته number چاپ کردن عدد
```

به جای رشته های آرایه ای که در زبان C مرسوم است از کلاس string، که یکی از کلاس های استاندارد C++ است که با نام STL (Standard Template Library) شناخته می شوند. بعدا در مورد این کتابخانه بیشتر توضیح خواهیم داد.

برای اینکه هنگام استفاده از عملگر های << و >> دچار اشتباه نشوید کافی است که cin >> a را اینگونه تفصیر کنید که جریان اطلاعات از cin به سمت a خواهد بود و بنابراین باید از << استفاده کنیم. به همین ترتیب در cout << a جریان اطلاعات از متغیر a به سمت cout خواهد بود، پس ا ز >> استفاده می کنیم.

همان طور که در قطع کد بالا مشاهده می کنید، cin و cout را به صورت ترکیبی نیز می توان به کار برد. قطعه کد زیر معادل کد بالا است:

```
int number;
string word;
cin >> number;
cin >> word;
cout << number;
cout << " ";
cout << word;
cout << endl;          // cout << '\n' --→ buffer را خالی نمی کند
```

مثال:

Standard Input	Standard Output
Someone 2 3	Someone says: 2 + 3 is: 5

```
#include <iostream>
#include <string>

using namespace std;

int main ()
{
        string str;
        int a, b;
        cin >> str >> a >> b;
        cout << str << " says: " << a << " + " << b;
        cout << " is: " << (a + b) << endl;
        return 0;
}
```

cin برای خواندن به این صورت عمل می کند که آن قدر فضای خالی (White-Space) از ورودی رد می کند تا به کاراکترهای متناسب با متغیری که قرار است مقداری در آن قرار دهد، برسد. سپس اقدام به ذخیره آن کاراکترها

می کند تا به یک کاراکتر فضای خالی برسد. سپس خواندن از ورودی را متوقف کرده و مقادیری را که خوانده است در متغیر مورد نظر ذخیره می کند. بعضی از کاراکتر های فضای خالی عبارتند از:

‘\t’ (Tab) ‘\n’ (New Line) ‘ ‘ (Space)

مثال:

Standard Input	Standard Output
Someone 2 3	Someone says: 2 + 3 is: 5

```
#include <iostream>
#include <string>

using namespace std;

int main ()
{
	string str;
	int a, b;
	cin >> str >> a >> b;
	cout << str << " says: " << a << " + " << b;
	cout << " is: " << (a + b) << endl;
	return 0;
}
```

C++ - getline

همان طور که اشاره شد، cin فضا های خالی را نادیده می گیرد و آن ها را رد می کند. حال اگر بخواهیم **یک خط** شامل فضا های خالی (به جز ‘\n’) را در زبان C++ بخوانیم، از تابع getline استفاده می کنیم. به صورت زیر:

```
string str;
getline (cin, str);    // str خواندن کل خط و ریختن آن در رشته
```

مثال:

Standard Input	Standard Output
Someone Else	Someone Else

اگر ورودی را با cin بخوانیم، فقط کلمه "Someone" را خوانده و آن را در خروجی چاپ می کند:

```
#include <iostream>
#include <string>

using namespace std;

int main ()
{
	string str;
	cin >> str;
	cout << str << endl;
	return 0;
}
```

اما اگر بخواهیم کل "Someone Else" را بخوانیم، از getline استفاده می کنیم:

```
#include <iostream>
#include <string>

using namespace std;

int main ()
{
	string str;
	getline (cin, str);
	cout << str << endl;
	return 0;
}
```

*نکته خیلی مهم: اگر از getline بعد از cin استفاده می کنید، آخرین کاراکتر '\n' (New Line) قبل از getline را رد کنید.

برای مثال:

Standard Input	Standard Output
7 Someone Else	Someone Else says: 7 is a lucky number!

```
#include <iostream>
#include <string>

using namespace std;

int main ()
{
	int a;
	string str;
	cin >> a;
	getline (cin, str);
	cout << str << " says: " << a << " is a lucky number!" << endl;
	return 0;
}
```

برنامه فوق خروجی " says: 7 is a lucky number!" را چاپ می کند؛ یعنی رشته مورد نظر ما به صورت "" خوانده شده که علت آن نادیده نگرفتن ادامه خط قبل و کاراکتر خط جدید بعد از cin می باشد. یعنی getline بعد از cin، در ادامه خط قبل، بلافاصله به کاراکتر خط جدید برخورد می کند و آن را اتمام خط تلقی کرده و خواندن را متوقف می کند. پس رشته ما "" خواهد بود.

برای نادیده گرفتن کاراکتر خط جدید می توان به دو روش عمل کرد:

۱- با یک getline به طور مجزا ادامه خط قبل را خوانده و کاراکتر خط جدید را رد کنیم و سپس با getline دیگری رشته مورد نظر را بخوانیم. به صورت زیر:

```
#include <iostream>
#include <string>

using namespace std;

int main ()
{
	int a;
	string str;
	cin >> a;
	getline (cin, str);  // کاراکتر خط جدید را رد می کند
	getline (cin, str);  // رشته مورد نظر را می خواند
	cout << str << " says: " << a << " is a lucky number!" << endl;
	return 0;
}
```

۲- از cin.ignore() استفاده کنیم. به صورت زیر:

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
	int a;
	string str;
	cin >> a;
	cin.ignore();
	getline (cin, str);
	cout << str << " says: " << a << " is a lucky number!" << endl;
	return 0;
}
```

C++ - freopen

گاهی در مسابقات ACM ممکن است تعداد TestCase های ورودی سوال خیلی زیاد باشند (مثلا در مورد سوالات هندسی و نظریه گراف). در این مورد برای تست کردن درستی الگوریتم خود ممکن است چندین بار نیاز باشد تا TestCase های نمونه سوال را امتحان کنید که این کاری طاقت فرسا و البته زمان بر است. برای اینکه در زمان صرفه جویی کنید، کافی است **فقط یک بار** ورودی های نمونه را که در سوال مطرح شده اند و همچنین TestCase هایی که خودمان در می آوریم را در یک فایل متنی ذخیره کنیم و در کد خود مشخص کنیم که به جای آنکه ورودی را از stdin بخواند، از فایل ورودی ما بخواند (یعنی به فایل به دید stdin نگاه کند). برای این کار از freopen استفاده می کنیم. به صورت زیر:

ابتدا خط زیر را در ابتدا کد خود اضافه می کنیم تا بتوانیم از freopen استفاده کنیم:

```
#include <cstdio>	// #include <stdio.h>
```

سپس کد زیر را برای خواندن از فایل اضافه می کنیم:

```
freopen( "آدرس فایل", "r", "stdin");
```

این خط باعث می شود که کد ما از فایل داده شده به عنوان stdin استفاده کرده و ورودی را از آن بخواند و همچنین خط زیر باعث می شود که کد ما از فایل داده شده به عنوان stdout استفاده کرده و خروجی خود را در آن بنویسد:

```
freopen( "آدرس فایل", "w", "stdout")
```

نکته: برای داوری در مسابقات برنامه نویسی ACM از همین روش استفاده می گردد.

C - scanf/printf

همان طور قبل تر گفته شد، در مسابقات ACM **بعضی اوقات ممکن است**، تعداد ورودی های یک سوال خیلی زیاد و یا ابعاد ورودی خیلی حجیم باشد. از آن جایی که cin/cout کند می باشند، استفاده از آن ها موجب می شود که زمان تعیین شده برای سوال را رد کنیم (TLE = Time Limit Error)، از این جهت از ورودی و خروجی زبان C، که خیلی سریع تر از cin/cout می باشند، استفاده می کنیم. از scanf برای خواندن ورودی و از printf برای نمایش خروجی استفاده می کنیم. برای آشنایی با آن ها می توانید از لینک های زیر استفاده کنید:

scanf: http://www.cplusplus.com/reference/cstdio/scanf/
printf: http://www.cplusplus.com/reference/cstdio/printf/

Java - Scanner/System.out

برای خواندن ورودی در زبان برنامه نویسی Java توصیه می شود که از کلاس Scanner استفاده نمایید. برای استفاده از آن دو خط زیر به ابتدای کد خود اضافه نمایید:

```
import java.util.*;
import java.io.*;
```

قبل از خواندن ورودی با Java ابتدا باید از کلاس Scanner یک Object در تابع main بسازید. به صورت زیر:

```
Scanner in = new Scanner(System.in);
```

فرض کنید می خواهیم یک عدد صحیح، یک کاراکتر و یک کلمه را از ورودی بخوانیم و بعد به خط بعد برویم:

```
int d = in.nextInt();
char c = in.next().charAt(0);
s1 = in.next();
in.nextLine();                // برای نادیده گرفتن ادامه خط
s2 = in.nextLine();
```

حال برای نشان دادن خروجی کافی است به روش زیر عمل کنید:

```
System.out.println(d + " " + c + " " + s1);
System.out.println(s2);
```

نکته: برای جلوگیری از نشان دادن یک خط جدید از کد زیر استفاده کنید:

```
System.out.print("Blah");
```

بررسی ورودی/خروجی در مسابقات ACM

حال به بررسی ورودی های معمول در سوال های مسابقات برنامه نویسی می پردازیم. به طور کلی ورودی سوالات اغلب یکی از سه حالت زیر است:

۱- تعداد ورودی ها و TestCase ها به وضوح گفته می شود، که برای خواندن ورودی در این روش کافی است از حلقه جهت دریافت ورودی ها استفاده کنیم. فرض کنید n تعداد ورودی ها باشد که در سوال داده شده است. به صورت زیر عمل می کنیم:

```
int n;
cin >> n;
for (int I = 1; I <= n; I++)            //for (int I = 0; I < n; I++)
{
        // هر ورودی را می خوانیم
        // الگوریتم خود را روی آن اعمال می کنیم
        // خروجی را چاپ می کنیم
}
```

و یا:

```
int n;
cin >> n;
int I = 1;                              // int I = 0;
while (I <= n)                          // while (I < n)
{
        // هر ورودی را می خوانیم
        // الگوریتم خود را روی آن اعمال می کنیم
        // خروجی را چاپ می کنیم
        I++;
}
```

۲- گاهی اوقات ورودی یک سوال کل فایل ورودی می باشد و ما باید تا آخر فایل ورودی (EOF = End Of File)، ورودی بخوانیم. فرض کنید n ورودی سوال است. برای خواندن ورودی تا آخر فایل به روش زیر عمل می کنیم:

```
int n;
while (cin >> n)        // تا زمانی که ورودی داریم و ورودی وارد می شود
{
        // الگوریتم خود را روی آن اعمال می کنیم
        // خروجی را چاپ می کنیم
}
```

و یا فرض کنید ورودی سوال یک رشته دلخواه است:

```
string str;
while (getline(cin, str))   // تا زمانی که ورودی داریم و ورودی وارد می شود
{
        // الگوریتم خود را روی آن اعمال می کنیم
        // خروجی را چاپ می کنیم
}
```

*نکته مهم: توصیه می شود که تا وقتی که **مجبور نیستید**، از eof استفاده نکنید؛ زیرا بعضی از ورودی ها ممکن است فضا های خالی اضافی داشته باشند.

۳- ممکن است در صورت سوال گفته شود " تا زمانی که به یک ورودی خاص نرسیدید، ورودی بخوانید"؛ یعنی در این مورد شرط پایان ورودی، رسیدن به یک ورودی خاص، با یک شرط خاص است. برای مثال فرض کنید در یک سوال، عدد n به عنوان ورودی باشد و شرط $n == 0$ به عنوان شرط پایان ورودی باشد؛ یعنی تا زمانی که n عددی مخالف 0 است ورودی بخوانیم. به این صورت عمل می کنیم:

```
int n;
while (cin >> n && n != 0)                    // while (cin >> n && n)
{
        // الگوریتم خود را روی آن اعمال می کنیم
        // خروجی را چاپ می کنیم
}
```

*نکته مهم: برای نشان دادن عدد صحیح n با ۳ رقم اعشار از کد زیر استفاده می کنیم:

```
cout.setf(ios::fixed);        // ios::fixed --→ عدد به صورت غیر علمی
cout.setf(ios::showpoint);    // ios::showpoint --→ نمایش نقطه اعشاری
cout.precision(3);            // نمایش سه رقم بعد از اعشار
cout << n << endl;
```

و یا می توانیم به جای خط اول و دوم کد بالا از کد زیر استفاده کنید:

```
cout.flags(ios::fixed | ios::showpoint);
```

همچنین می توانید از کلاس iomanip استفاده نمایید که برای این کار ابتدا این کلاس را در ابتدای کد خود به صورت زیر include می کنیم:

```
#include <iomanip>
```

و سپس خط زیر را که معادل قطعه کد بالا می باشد را اضافه می کنیم:

```
cout << fixed << showpoint << setprecision(3) << n << endl;
```

علاوه بر روش های فوق می توانید از printf نیز به همین منظور استفاده کنید.

تمرین

- اعداد صحیح n و m را تا آخر فایل ورودی خوانده و مجموع هر یک از آن ها را در یک خط مجزا چاپ کنید.
- تا انتهای فایل رشته های مختلف را خوانده و آخرین کاراکتر هر رشته را در یک خط به طور مجزا چاپ کنید.
- اعداد صحیح n را تا انتهای فایل خوانده و دو رقم آخر آن ها را در یک خط مجزا و با یک فاصله از هم چاپ کنید.
- اعداد صحیح n را از ورودی خوانده و حاصل n / 7 را تا دو رقم اعشار در خروجی نمایش دهید. همچنین **بین هر دو خط متوالی** در خروجی یک خط خالی چاپ کنید.

پاسخ تمرینات فوق را به زبان C++ و یا Java و در غالب تنها یک فایل متنی که در هر تمرین به صورت comment شده در آن می باشد به آدرس acmparand@gmail.com ارسال نمایید.
همچنین می توانید سوالات احتمالی خود را در گروه مطرح نمایید تا به آن ها پاسخ داده شود.