دستورات ورودی و خروجی

اشیای cout و cin در C++ برای ارسال خروجی به صفحه نمایش و دریافت مقادیر از صفحه کلید استفاده می شوند. این توابع در فایل هدری به نام iostream قرار دارد که باید ضمیمه شود. کلیه نیازهای ورودی و خروجی با این اشیاء برطرف می شود.

با وجودیکه C++ سازگاری خود را با C حفظ کرده است و می توان از توابع printf() و scanf() در کتابخانه <stdio.h> استفاده کرد، اشیای ورودی/خروجی دیگری را فراهم آورده است که بسیار قوی تر و مطمئن تر است. کلاس iostream، موجود در کتابخانه ای با همین نام، اشیاء و متدهای موردنیاز را در اختیار می گذارد. این کتابخانه از ostream (برای خروجی) و istream (برای ورودی) مشتق می شود.

ورودی/خروجی به صورت جریانی (stream) از کاراکترها یا بایت ها از برنامه به صفحه نمایش، چاپگر یا فایل به عنوان خروجی و یا از صفحه کلید به عنوان ورودی تصور می شود. جریان ها راه انعطاف پذیری برای کار با ورودی و خروجی است که درمورد فایل های I/O هم صدق می کنند.

cout

شئ cout متعلق به کتابخانه iostream برای نمایش داده روی صفحه نمایش است. cout به کامپایلر فرمان می دهد که جریان بایت ها را به دستگاه خروجي پیش فرض که معمولا مانیتور است هدایت کند.

cout مخفف console output است.

بعد از cout علامت >> قرار می گیرد که جهت متن ارسالی را تعیین می کند. در C سمبل >> برای شیفت بیت های عدد صحیح به سمت چپ بکار می رفت. در C++ این سمبل برای انواع داده مانند int، float و رشته پشتیبانی می شود، چند مقدار با >> بهم ریسمان می شود تا خروجی ارسال شود.

مثال.

```
cout << "Some Text" << intvalue << floatdouble;
```

این گرامر خاص ممکن است زیرا هر >> درواقع تابعی را فراخوانی می کند که ارجاعی به یک شئ ostream را بر می گرداند بنابراین خط بالا درواقع اینکار را می کند.

cout.<<("some text").cout.<<(intvalue).cout.<<(floatdouble).cout;

برای اینکه پیغام در خط جداگانه ای نمایش داده شود \n را اضافه کنید. \n به کامپایلر می گوید از یک خط جدید شروع کند. \n یک نوع Escape Code است که برای فرمت بندی خروجی استفاده می شوند.

مثال. نمایش متن "Hello World!" روی مانیتور.

```
#include <iostream>
int main()
{
    //print hello word on the screen
    cout << "Hello World!\n";
    return 0;
}
```

بعضی از کدهای escape در جدول زیر آمده است.

کدهای escape

Escape Code	شرح
\n	New line
\r	Carriage retuen
\b	Backspace
\t	Horizontal tab
\'	Single quotation mark
\f	Form feed
\a	Bell (alert)
\v	Vertical tab
\\	Backslash
\"	Double quotation mark
\?	Literal question mark

نکته. بیشتر از یک escape code می تواند در متن خروجی باشد.
نکته. اگر در یک رشته بخواهیم علامت گیومه (") قرار دهیم به معني پایان رشته تعبیر می شود بنابراین باید از یک Escape code استفاده شود.

مثال. در برنامه زیر عبارت" escape keys " درون گیومه و عبارت ' useful ' درون تک گیومه نمایش داده می شوند.

```
#include <iostream>
int main()
{
  cout << "As you can see these \" escape keys \" \n";
  cout << "are quite \'useful \' \a \\ in your code \\ \a \n";
  return 0;
}
```

برای تعیین شروع خط جدید بجای \n دستور endl می تواند در انتهای خروجی ذکر شود. endl باعث خالی شدن بافر C++ می شود.

مثال.

```
#include <iostream>
int main()
{
  cout << "You have previously used the \\n key to get a new line \n";
  cout << "However you can also use the endl command"<<endl;
  return 0;
}
```

قالب بندی خروجی

وقتی عددی نمایش داده می شود صفات زیر مورد نظر است:

- مقدار فضای مورد نیاز برای عدد روی صفحه
- چپ چین یا راست چین بودن (اعداد تمایل به راست چین دارند)
- تعداد ارقام اعشار
- علامت برای اعداد منفی
- نمای علمی برای اعداد بزرگ

این صفات توسط شئ cout و توابع کتابخانهiomanip قابل تنظیم هستند.

در C با استفاده از فرامین فرمت مانند %d در تابع printf() قادر به فرمت بندی خروجی هستیم. در C++ فرمت بندی خروجی به طریق متفاوتی از طریق اضافه کردن دستکاری کننده ها (manipulators) به جریان خروجی انجام می گیرد.

manipulator تابعی است که می تواند خصوصیات جریان خروجی را تغییر بدهد. manipulator می توانند به جریان های ورودی یا خروجی اضافه شوند.

مثال.endl خط را تمام می کند و خط جدید را شروع می کند.

```
count << endl;
cout << "Some Text" << endl << endl; // Two blank lines
```

تابع به صورت زیر هم می تواند استفاده شود.

```
endl(cout);
```

کلاس ostream که برای خروجی استفاده می شود خود از کلاس ios مشتق شده است که از ios_base گرفته شده است. کلاس اخیر توابع عمومی را برای دستکاری کننده ها تعریف می کند.

کلاس iostream از ostream و istream ارث بری دارد بنابراین مثال های cout می توانند iostream را استفاده کنند.

انواع دستکاری کننده ها

اکثر دستکاری کننده ها در <ios.h> اعلان شده اند اما endl، ends و flush از <ostream.h> می آیند. بعضی یک پارامتر می گیرند و از <iomanip.h> می آیند.

دستکاری کننده های کتابخانه <ostream.h>:

- endl – خط را خاتمه داده و flush را صدا می زند.
- ends – کاراکتر '\0'(NULL) را در جریان درج می کند.
- flush – بافر را بلافاصله به خروجی می فرستد.

دستکاری کننده های کتابخانه<ios> که بیشتر آنها در <ios_base> اعلان شده اند:

- boolalpha – درج یا استخراج اشیای بولین مانند "true" و "false".
- noboolalpha – درج یا استخراج اشیای بولین مانند مقادیر عددی.

- fixed – درج مقادیر ممیزشناور در فرمت ثابت.
- scientific – درج مقادیر ممیزشناور در فرمت علمی.

- Internal-justify – internal
- Left-justify – left
- Right-justify – right

- dec – درج یا استخراج مقادیر صحیح در فرمت دهدهی.
- hex – درج یا استخراج مقادیر صحیح در فرمت هگز.
- oct – درج یا استخراج مقادیر صحیح در فرمت اکتال.

- showbase – مبنای مقدار را پیشوند آن می کند.
- noshowbase – مبنای مقدار را پیشوند نمی کند.
- showpoint – همیشه نقطه اعشار را هنگام درج مقدار اعشار نمایش می دهد.
- noshowpoint – نقطه اعشار را اگر لازم نیست نمایش نمی دهد.
- showpos – اگر عدد مثبت است علامت بعلاوه (+) را درج می کند.
- noshowpos – اگر عدد مثبت است علامت بعلاوه (+) را درج نمی کند.
- skipws – از روی فاصله ها می پرد.
- noskipws – از روی فاصله ها نمی پرد.
- uppercase – حروف کوچک را با حروف بزرگ جایگزین می کند. تنها روی خروجی تولید شده مانند اعداد هگز تاثیر دارد.

- nouppercase – حروف کوچک را با حروف بزرگ جایگزین نمی کند.
- unitbuf – بافر را بعد از درج خالی می کند.
- nounitbuf – با را بعد از هر درج خالی نمی کند.

اکثر این دستکاری کننده ها یک بیت در فلگ را تنظیم می کنند. یک فلگ مقداری است که برای تعیین یک سری از کارها استعمال می شود. فلگ ها معمولا مقادیر صحیح هستند اما اغلب دارای یک نام یا برچسب هستند تا این مقدار خواناتر باشد.

تنظیم فلگ مستقیما توسط cout.setf() قابل انجام است و توسط cout.unsetf() خنثی می شود. این توابع متعلق به کتابخانه <iomanip> هستند.

تابع setf به دو صورت می تواند استفاده شود:

```
setf( flagvalues);
setf( flagvalues, maskvalues);
unsetf( flagvalues);
```

مثال. برای قالب بندی خروجی به صورت scientific، uppercase و boolalpha پارامترها برای تنظیم بیت ها به صورت زیر ارسال می شود:

```
cout.setf( ios_base::scientific | ios_base::uppercase | ios_base::boolalpha) ;
cout << hex << endl;
cout << 1234 << endl;
cout << dec << endl;
cout << 123400003744.98765 << endl;
bool value=true;
cout << value << endl;
cout.unsetf( ios_base::boolalpha) ;
cout << value << endl;
```

خروجی به صورت زیر تولید می شود:

```
4D2
1.234000E+011
true
1
```

هر کاراکتری را می توان توسط casting به خروجی ارسال کرد. برای مثال Char(27) کاراکتر escape را ارسال می کند.

اگر دو ثابت رشته در جوار هم باشند و بین آنها علامتی نباشد کامپایلر دو رشته را بهم می چسباند.

مثال.

```
#include <iostream>

int main() {
  cout << "This is far too long to put on a "
    "single line but it can be broken up with "
    "no ill effects\nas long as there is no "
    "punctuation separating adjacent character "
    "arrays.\n";
}
```

cerr و clog

cerr و clog مشابه cout دو شئ تعریف شده در ostream هستند.

مثال. برنامه زیر نشان می دهد چگونه cerr می تواند به جای cout بکار برود.

```
#include <iostream>

int  main()
{
   cerr.width(15) ;
   cerr.right;
   cerr << "Error" << endl;
   return 0;
}
```

ساختن log از رویدادهای برنامه روش خوبی برای تعیین اشکالات آن است. log بعد از هر فراخوانی روی دیسک خالی می شود بنابراین اگر رویدادی باعث سقط برنامه شود بلافاصله بعد از سقط می توان log را مشاهده کرد.

cout و clog هردو خروجی را بافر می کنند یعنی ابتدا کل خروجی در بافر ذخیره می شود سپس همگی یکباره با هم به خروجی فرستاده می شود. cerr خروجی را بافر نمی کند و آنرا بلافاصله به دستگاه خروجی می فرستد.

اشکال مهم بافر کردن این است که اگر برنامه سقط کند محتوای بافر از دست می رود و تشخیص علت سقط برنامه سخت تر می شود.

cin

کلاس iostream امکان خواندن ورودی را هم فراهم می کنند. شئ مورد استفاده برای ورودی استاندارد cin است که ورودی را معمولا از کنسول می گیرد اما می تواند از منابع دیگر هم دریافت کند.

عملگری که همراه با cin استفاده می شود >> است که برای دریافت ورودی از نوع آرگومانش صبر می کند. مثلا اگر آرگومان صحیح باشد برای یک عدد صحیح از صفحه کلید منتظر می ماند.

مثال. برنامه زیر یک عدد صحیح را دریافت کرده معادل اکتال و هگز آنرا نمایش می دهد.

```
#include <iostream>

int main() {
  int number;
  cout << "Enter a decimal number: ";
  cin >> number;
  cout << "value in octal = 0"
       << oct << number << endl;
  cout << "value in hex = 0x"
       << hex << number << endl;
}
```

تابع cin متوجه نوع متغیر می شود و داده ورودی را به نوع متناسب تبدیل می کند.

مثال. برنامه زیر تابع cin را برای خواندن سه عدد که با space از هم جدا می شوند استفاده می کند. بعد از وارد کردن مقادیر باید کلید enter را فشار داد.

```
#include <iostream>

int main()
{
   int a = 0;
   float b = 0.0;
   int c = 0;
   cout << "Please Enter an int, a float and int separated by spaces" <<endl;
   cin >> a >> b >> c;
   cout << "You entered " << a << " " << b << " " << c << endl;
   return 0;
}
```

اگر 3 7.2 3 وارد شود خروجی به صورت "You entered 3 7.2 3" خواهد بود.

اگر 8 5 3.76 وارد شود خروجی به صورت "You entered 3 0.76 5 " می شود. چون نقطه ممیز جز عدد صحیح نیست به عنوان شروع عدد اعشاری درنظرگرفته می شود و بقیه خط ورودی از بین می رود.

اگر ورودی به طور موفق تبدیل نشود شئ cin یک بیت شکست را تنظیم می کند که قسمتی از ios است و توسط تابع fail() هم در cin و هم در cout به صورت زیر قابل خواندن است.

```
if (cin.fail() ) //do something
```

برای صفحه نمایش روشن است که cout.fail() بندرت ممکن است استفاده شود اما درفایل های I/O می تواند بکار برود.

تابع good() هم برای cin و cout وجود دارد.

cin بهترین دستور برای همه شرایط نیست اما ساده ترین را ه است. تابع get یک کاراکتر تکی صرفنظر از نوع را دریافت می کند.

تابع getline برای دریافت تعداد معینی کاراکتر است. این تابع وقتی می خواهید داده ای را درون آرایه ذخیره کنید سودمند است. cin اجازه می دهد کاربر داده های بیشتری را در آرایه ذخیره کند که overloading نام دارد. توسط تابع getline تعداد بایت هایی که می خواهید کاربر وارد کند را تعیین می کنید. هرکاراکتر که کاربر تایپ می کند یک بایت فضا می برد. مثلا اگر برای تابع ۴بایت تعریف کنید و کاربر کلمه Computer را وارد کند تنها Comp دریافت می شود.

مثال.

```
#include <iostream>

int main()
{
  char text[10];

  cout << "Please enter a word\n";
  cin.getline(text,10);
  cout << text << endl;

  return 0;
}
```

کلیه ورودی هائی که با cin خوانده می شوند نیاز دارند کلید Enter یا Return زده شود. در C++ استاندارد راهی برای خواندن مستقیم کلیدها از صفحه کلید وجود ندارد. اما می توان از کتابخانه های دیگر استفاده کرد.