如何向计算机输入及输出我们所需要的信息呢

新知一下

逻辑航线

标准输入输出

NOI基础语法系列课程

版本：2.0.0

讲师：孙伟航

01

标准输入输出
stdin / stdout

标准输入输出

stdin：是标准输入，一般指键盘输入到缓冲区里的东西。简单的说，它是一个专用的文件句柄，我们可以向它请求我们需要的输入数据。

stdout：输出方式是行缓冲。输出的字符会先存放在缓冲区，等按下回车键时才进行实际的I/O操作。

02 基本输入输出流

iostream

iostream

该文件定义了 cin、cout对象，分别对应于标准输入流、标准输出流、非缓冲标准错误流和缓冲标准错误流。

用法：`#include <iostream>`

cout

逻辑航线

示例:

```
#include <iostream>
using namespace std;
int main( )
{
        char a = “c”;
        cout << "Value of str is : " << a << endl;
}
```

输出

```
Value of str is : c
```

cout

常见标志：

标志	功能
boolalpha	可以使用单词”true”和”false”进行输入/输出的布尔值.
oct	用八进制格式显示数值.
dec	用十进制格式显示数值.
hex	用十六进制格式显示数值.
left	输出调整为左对齐.
right	输出调整为右对齐.
scientific	用科学记数法显示浮点数.
fixed	用正常的记数方法显示浮点数(与科学计数法相对应).
showbase	输出时显示所有数值的基数.
showpoint	显示小数点和额外的零，即使不需要.
showpos	在非负数值前面显示” + （正号）”.
skipws	当从一个流进行读取时，跳过空白字符(spaces, tabs, newlines).
unitbuf	在每次插入以后，清空缓冲区.
internal	将填充字符回到符号和数值之间.
uppercase	以大写的形式显示科学记数法中的”e”和十六进制格式的”x”.

cout

示例，将10000按照八进制进行打印：

```
#include <iostream>
#include <iomanip>


using namespace std;


int main(void)
{
    long n = 10000;
    cout << hex << n ;
    return 0;
}
```

cin

功能1：接受单个字符，数字，不可接受空白字符
功能2：接受一个字符数组，遇“空格”、“Tab”、“回车”结束，后面的字符不再读取。

原理：

cin读取数据也是从缓冲区中获取数据，缓冲区为空时，cin的成员函数会阻塞等待数据的到来，一旦缓冲区中有数据，就触发cin的成员函数去读取数据。

当cin>>从缓冲区中读取数据时，若缓冲区中第一个字符是空格、tab或换行这些分隔符时，cin>>会将其忽略并清除，继续读取下一个字符，若缓冲区为空，则继续等待。

但是如果读取成功，字符后面的分隔符是残留在缓冲区的，cin>>不做处理。

cin

使用cin读取输入100的过程如下：

我们先通过键盘输入100，缓冲区如下：

缓冲区			
1	0	0	\n

之后，我们调用cin>>a，将数据赋给a，此时缓冲区如下

缓冲区			
\n			

cin

逻辑航线

示例1：

```
#include <iostream>
using namespace std;
int main( )
{
    int a, b;
    char c, d;
    cin >> a >> b >> c >> d;
    cout << "a:" << a << endl;
    cout << "b:" << b << endl;
    cout << "c:" << c << endl;
    cout << "d:" << d << endl;
}
```

结果1：

```
1 2 a b
a:1
b:2
c:a
d:b
```

结果2：

```
1 2
```

不输入其他字符，只输入空格和回车，程序不能终止

cin

逻辑航线

示例2：

```
#include <iostream>
using namespace std;
int main( )
{
        char name[50];
        cout << "请输入您的名称：  ";
        cin >> name;
        cout << "您的名称是：  " << name << endl;
}
```

结果：

```
请输入您的名称：  addffd 121d
您的名称是：  addffd
```

注意：空格后的数字没有被获取

cin

逻辑航线

示例3：

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


using namespace std;
int main()
{
    char c;
    cin >> c;
    cout << c;
    cout<<endl<<"RETURN END"<<endl;
    return 0;
}
```

结果：

```
            aadfsac
    a
    RETURN END
```

注意：自动忽略了前面的空格

cin

C++ 编译器根据要输入值的数据类型，选择合适的流提取运算符来提取值，并把它存储在给定的变量中。流提取运算符 >> 在一个语句中可以多次使用，如果要求输入多个数据，可以使用如下语句：

```
cin >> name >> age;
```

这相当于下面两个语句：

```
cin >> name;
cin >> age;
```

cin.get()

功能1：接受单个字符，数字，空白字符
功能2：cin.get(字符数组名，接收字符数)用来接收一行字符串，可以接收除回车外的字符。
该函数以换行符结束，但之后会丢弃换行符并以'\0'代替

示例1：

```
#include <iostream>
using namespace std;
int main( )
{
    int a, b;
    char c, d;
    a = cin.get();
    b = cin.get();
    c = cin.get();
    d = cin.get();
    cout << "a:" << a << endl;
    cout << "b:" << b << endl;
    cout << "c:" << c << endl;
    cout << "d:" << d << endl;
}
```

结果：

```
1 2 a
a:49
b:32
c:2
d:
```

不输入其他字符，只输入空格和回车，程序正常终止

cin.get()

逻辑航线

示例1：

```
#include <iostream>
using namespace std;
int main( )
{
    char c[5];
    cin.get(c,5); //数组被填满，或者遇回车结束
    for(int i=0; i<5; i++)
    {
    cout << c[i];
    }
}
```

结果：

```
12aabedf
12aa
```

cin.getline()

该函数用法等同cin.get()

03 cin常见异常

cin与cin.get()混用产生的异常

cin与cin.get()混用时，会产生异常现象，示例：

```
#include <iostream>
using namespace std;
int main( )
{
    char ch; //定义一个字符变量
    int number; //定义一个整型变量
    cout << "Enter a number: ";
    cin >> number; // 读取整数
    cout << "Enter a character: ";
    ch = cin.get() ; // 读取字符
    cout << "Thank You!\n";
}
```

结果：

```
Enter a number: 1
Enter a character:
Thank You!
```

问题：我们可以看到，程序运行时我们并没有输入任何字符串，该程序就自动终止了，这是为什么呢？

cin与cin.get()混用产生的异常

原因如下：

我们第一次输入了数字100，之后按下了回车，所以键盘缓冲区就存下了这样的数据。

缓冲区			
1	0	0	\n

cin不读取换行，所以当它读取完100后，缓冲区只剩下了\n，如下图：

缓冲区			
\n			

这时下一句的cin开始执行，它又遇到了\n，以为是我们让它终止。所以，程序就自动终止了。

cin与cin.get()混用产生的异常

逻辑航线

我们可以使用cin.ignore对其进行修正

```
#include <iostream>
using namespace std;
int main( )
{
    char ch; //定义一个字符变量
    int number; //定义一个整型变量
    cout << "Enter a number: ";
    cin >> number; // 读取整数
    cout << "Enter a character: ";
    cin.ignore();  //修正
    ch = cin.get() ; // 读取字符
    cout << "Thank You!\n";
}
```

正确结果

```
Enter a number: 1
Enter a character: 3
Thank You!
```

iomanip

该文件通过所谓的参数化的流操纵器（比如 **setw** 和 **setprecision**），来声明对执行标准化 I/O 有用的服务。

用法：`#include <iomanip>`

iomanip 中定义的操作符：

操作符	描述	输入	输出
resetiosflags(long f)	关闭被指定为f的标志	√	√
setbase(int base)	设置数值的基本数为base		√
setfill(int ch)	设置填充字符为ch		√
setiosflags(long f)	启用指定为f的标志	√	√
setprecision(int p)	设置数值的精度(四舍五入)		√
setw(int w)	设置域宽度为w		√

iomanip

示例代码：

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(void)
{
    cout<<setbase(8)<<setw(8)<<255<<endl;
    cout<<"00000000";
    return 0;
}
```

输出：

```
     377
00000000
```

04 字符输入输出

iostream

getchar

逻辑航线

说明： 接受从键盘输入的单个字符

注意： 1、只能接受一个任意字符（包括空白），如果输入了多个字符，则只取第一个
2、输入的数字也按照字符处理
3、该函数会等待用户输入，直至按下回车。（可用于暂停程序）
4、当使用多个getchar方法来接受多个字符的时候，应该一次性输入所需字符，最后再按回车。否则会把回车作为一个字符传给后面的函数

示例

```
#include <iostream>
using namespace std;
int main(int argc, char** argv)
{
    char ch = getchar();
    cout << "input=" << ch << endl;
    return 0;
}
```

结果

输入：adfa
输出：input=a

putchar

逻辑航线

说明： 向标准输出输出单个字符

示例

```
#include <iostream>
using namespace std;
int main(int argc, char** argv)
{
    char ch = getchar();
    putchar(ch);
    putchar(66); //用10进制ASCII码输出字母B
    return 0;
}
```

结果

输入：adfa
输出：aB

05 字符串输入

string

getline

格式： getline(ifstream& input, string s, char delimitChar)

说明： **input：**是输入的对象，可以是一个文件，也可以是标准输入（cin）
s： 是接受字符串，所读取的信息存储在s中
delimitChar：是分隔符，默认是空白符

当函数读到分隔符或文件末尾时，就会停止，同时将该符号从缓存区中取走。遇到换行符时，会将其舍弃。

示例1：

```
#include <iostream>
#include <string>
using namespace std;
int main(int argc, char** argv)
{
    string s;
    getline(cin,s);
    cout << s << endl;
    return 0;
}
```

结果

```
输入：abc def ads
输出：abc def ads
```

getline

逻辑航线

示例2：

```
#include <iostream>
#include <string>
using namespace std;
int main(int argc, char** argv)
{
    string s;
    getline(cin,s,'#');
    cout << s << endl;
    return 0;
}
```

结果

输入： abc def #ads
输出： abc def

06 格式化输入输出

cstdio

scanf

逻辑航线

声明 `int printf(const char *format, ...)`

参数 `[=%[*][width][modifiers]type=]`

参数	描述
*	这是一个可选的星号，输入的值不赋给一个变量。
width	这指定了在当前读取操作中读取的最大字符数。
modifiers	修饰符，即使用“l”或者“h”
type	一个字符，指定了要被读取的数据类型以及数据读取方式。

scanf

可识别类型

类型	说明
d,i	用于十进制整数
u	以无符号十进制形式输入十进制整数
o	用于输入八进制整数
x	用于输入十六进制整数
c	用于输入单个字符
s	用于输入字符串（非空格开始、空格结束、字符串变量以'\0'结尾）
f	用于输入实数（小数或指数均可）
e	与f相同（可与f互换）
附加格式	说明
l	ld用于长整型，即long int
	lld用于超长整型，即long long int
	lf用于双精度浮点，即double
h	hd用于短整型，即short

scanf

*号用法:

```
#include <cstdio>
#include <iostream>
using namespace std;
int main(void)
{
    int a,b;
    scanf("%d%*d%d",&a,&b);
    cout<<"a:"<<a<<" b:"<<b<<endl;
    return 0;
}
```

输出:

```
1 2 3     //默认以空格作为结束符号
a:1 b:3
```

scanf

width用法:

```
#include <cstdio>
#include <iostream>
using namespace std;
int main(void)
{
    int a,b;
    scanf("%4d%4d",&a,&b);
    cout<<"a:"<<a<<" b:"<<b<<endl;
    return 0;
}
```

输出:

```
1234567
a:1234 b:567
```

scanf

间隔符用法:

```
#include <cstdio>
#include <iostream>
using namespace std;
int main(void)
{
    int a,b,c;
    scanf("%d,%d,%d",&a,&b,&c);
    cout<<"a:"<<a<<" b:"<<b<<" c:"<<c<<endl;
    return 0;
}
```

输出:

```
1,2,3     //注意此处必须输入逗号
a:1 b:2 c:3
```

scanf

间隔符用法:

```
#include <cstdio>
#include <iostream>
using namespace std;
int main(void)
{
    int a,b,c;
    scanf("%d,%d,%d",&a,&b,&c);
    cout<<"a:"<<a<<" b:"<<b<<" c:"<<c<<endl;
    return 0;
}
```

输出:

```
1,2,3     //注意此处必须输入逗号
a:1 b:2 c:3
```

printf

逻辑航线

声明 `int printf(const char *format, ...)`

参数 `[=%[flags][width][.precision][length]specifier =]`

flags（标识）	描述
-	在给定的字段宽度内左对齐，默认是右对齐（参见 width 子说明符）。
+	强制显示符号。
空格	如果没有写入任何符号，则在该值前面插入一个空格。
#	与 o、x 或 X 说明符一起使用时，非零值前面会分别显示 0、0x 或 0X。
	与 e、E 和 f 一起使用时，会强制输出包含一个小数点，即使后边没有数字时也会显示小数点。默认情况下，如果后边没有数字时候，不会显示显示小数点。
	与 g 或 G 一起使用时，结果与使用 e 或 E 时相同，但是尾部的零不会被移除。
0	在指定填充 padding 的数字左边放置零（0），而不是空格（参见 width 子说明符）。

printf

width（宽度）	描述
(number)	要输出的字符的最小数目。如果输出的值短于该数，结果会用空格填充。如果输出的值长于该数，结果不会被截断。
*	宽度在 format 字符串中未指定，但是会作为附加整数值参数放置于要被格式化的参数之前。

.precision（精度）	描述
.number	对于整数说明符（d、i、o、u、x、X）：precision 指定了要写入的数字的最小位数。如果写入的值短于该数，结果会用前导零来填充。如果写入的值长于该数，结果不会被截断。精度为 0 意味着不写入任何字符。
	对于 e、E 和 f 说明符：要在小数点后输出的小数位数。
	对于 g 和 G 说明符：要输出的最大有效位数。
	对于 s: 要输出的最大字符数。默认情况下，所有字符都会被输出，直到遇到末尾的空字符。
	对于 c 类型：没有任何影响。
	当未指定任何精度时，默认为 1。如果指定时不带有一个显式值，则假定为 0。
.*	精度在 format 字符串中未指定，但是会作为附加整数值参数放置于要被格式化的参数之前。

printf

length（长度）	描述
h	参数被解释为短整型或无符号短整型（仅适用于整数说明符：i、d、o、u、x 和 X）。
l	参数被解释为长整型或无符号长整型，适用于整数说明符（i、d、o、u、x 和 X）及说明符 c（表示一个宽字符）和 s（表示宽字符字符串）。
L	参数被解释为长双精度型（仅适用于浮点数说明符：e、E、f、g 和 G）。

printf

示例一：

```
#include <cstdio>
#include <iostream>
using namespace std;
int main(void)
{
    int i=1;
    long j=123;
    printf("%d,%2d,%03d,%1d,%-4d,%05ld",i,i,i,j,j,j);
    return 0;
}
```

输出：

```
1, 1,001,123,123 ,00123
```

printf

示例二：

```
#include <cstdio>
#include <iostream>
using namespace std;
int main(void)
{
    int i=-1;
    int j=100;
    printf("%+d,%+d",i,j);
    return 0;
}
```

输出：

```
-1,+100
```

printf

逻辑航线

示例三：

```
#include <cstdio>
#include <iostream>
using namespace std;
int main(void)
{
    float x = (float)102.1;
    printf("%6.2f\n", x);
    printf("%6.2f%6.2f\n", x,x);
    printf("%9.2f%9.2f\n", x,x);
    printf("%09d\n",0);
    printf("%-9.2f%-9.2f\n", x,x);
    printf("%09.2f\n", x);
    printf("%09.2f%09.2f\n", x,x);
    return 0;
}
```

输出：

```
102.10
102.10102.10
   102.10   102.10
000000000
102.10 102.10
000102.10
000102.10000102.10
```

07 字符串的特殊处理

scanf读取字符串

大数据的时候 cin cout 会超时，用scanf printf可以避免这个问题。
string 类型比 char 数组要方便很多，scanf也可以读取string但是需要预先分配好空间地址。

```
#include <iostream>
using namespace std;
int main( )
{
    string a;
    a.resize(100); //需要预先分配空间
    scanf("%s", &a[0]);
    cout << a;
}
```

printf输出字符串

printf输出字符串是针对char *的，换言之，printf只能输出C++语言中的内置数据，string不是c语言内置数据，所以不能简单的使用%s输出字符串，我们需要使用c_str或者cout方法。

```
#include <iostream>
using namespace std;
int main( )
{
    string ss = "abcddef adf";
    printf("%s\n", ss.c_str()); //方法一
    cout << ss; //方法二
}
```

08 特殊字符

换行连接符

字符串太长，换行显示，怎么办？

我们可以使用反斜杠“\”或双引号

```
#include <cstdio>
#include <iostream>
using namespace std;
int main(void)
{
    string str1 = "abcd\
    1234";
    string str2 = "abcd"
    "1234";
    cout << str1 << endl;
    cout << str2 << endl;
    return 0;
}
```

输出：

```
abcd1234
abcd1234
```

转义字符

转义字符是很多程序语言、数据格式和通信协议的形式文法的一部分。对于一个给定的字母表，一个转义字符的目的是开始一个字符序列，使得转义字符开头的该字符序列具有不同于该字符序列单独出现时的语义。因此转义字符开头的字符序列被叫做转义序列。

转义序列通常有两种功能。第一个是编码一个句法上的实体，如设备命令或者无法被字母表直接表示的特殊数据。第二种功能，也叫字符引用，用于表示无法在当前上下文中被键盘录入的字符（如字符串中的回车符），或者在当前上下文中会有不期望的含义的字符（如C语言字符串中的双引号字符"，不能直接出现，必须用转义序列表示）。

最直观的理解：转换了原有符号的意义。

转义字符

逻辑航线

转义字符	意义	ASCII码值（十进制）
\a	响铃(BEL)	7
\b	退格(BS)，将当前位置移到前一列	8
\f	换页(FF)，将当前位置移到下页开头	12
\n	换行(LF)，将当前位置移到下一行开头	10
\r	回车(CR)，将当前位置移到本行开头	13
\t	水平制表(HT) （跳到下一个TAB位置）	9
\v	垂直制表(VT)	11
\\	代表一个反斜线字符"\'	92
\'	代表一个单引号（撇号）字符	39
\"	代表一个双引号字符	34
\?	代表一个问号	63
\0	空字符(NULL)	0
\ddd	1到3位八进制数所代表的任意字符	三位八进制
\xhh	1到2位十六进制所代表的任意字符	二位十六进制

09

循环读取

循环读取

我们可以通过各类读取函数（cin,getline等）与while函数进行配合，以达到读取不定长度的信息。

循环读取字符与数字

逻辑航线

```
#include <iostream>
using namespace std;
int main()
{
    char c; //适用于数字，字符
    while(true)
    {
        c = cin.get();
        cout << c;
        //if(cin.peek() == '\n')   //两种判断方法
        if(c == '\n')
        {
                break;
        }
    }
    cout<<endl<<"cin char END"<<endl;
    return 0;
}
```

结果：

```
Look! There is a computer!
Look! There is a computer!
cin char END
```

循环读取字符串


```
#include <iostream>
#include<stdio.h>
#include<string>
using namespace std;
int main()
{
    string s; //字符串
    while(true)
    {
        cin>>s;
        cout << s;
        char c = cin.get();
        //if(cin.peek() == '\n')   //两种判断方法
        if(c == '\n')
        {
                break;
        }
    }
    cout<<endl<<"cin string END"<<endl;
    return 0;
}
```

结果：

```
Look! There is a computer!
Look!Thereisacomputer!
cin string END
```

10 重定向

重定向

想从某data.input文件中读入数据，而不是从小黑窗里键盘输入。或者想直接输出到某个data.txt文件，而不是打印在小黑窗里，就需要用到文件重定向。

```
#include <iostream>
#include <string>
#include <cstdio>
using namespace std;
int main()
{
    string str;
    freopen("input.txt","r",stdin); //打开输入文件
    freopen("output.txt","w",stdout); //打开输出文件
    while(getline(cin,str))
    {
        cout << str << endl;
    }
    cout << "The end";
    fclose(stdin);//关闭文件
    fclose(stdout);//关闭文件
    return 0;
}
```

上机练习

逻辑航线

1、完成在线OJ　http://noi.openjudge.cn/ch0101/

2、测试cout,scanf,printf,iomanip各个方法及参数

易错点

printf可以使用%f打印双精度

scanf必须使用%lf来接受双精度

保险起见，只要遇到双精度，就使用%lf

感　谢　观　看

联系地址：河北省廊坊市大厂县孔雀英国宫二期