

Практическая 0

ПОТОЧНЫЙ ВВОД-ВЫВОД

ПОТОКИ

- Стандартные потоки (istream, ostream, iostream) служат для работы с терминалом.
- Строковые потоки (istrstream, ostrstream, strstream) служат для ввода-вывода из строковых буферов, размещенных в памяти.
- Файловые потоки (ifstream, ofstream, fstream) служат для работы с файлами.

ios базовый потоковый класс

streambuf буферизация потоков

istream ПОТОКИ ВВОДА

ostream ПОТОКИ ВЫВОДА

iostream двунаправленные потоки

iostream_withassign ПОТОК с

переопределенной операцией присваивания

istrstream строковые потоки ввода

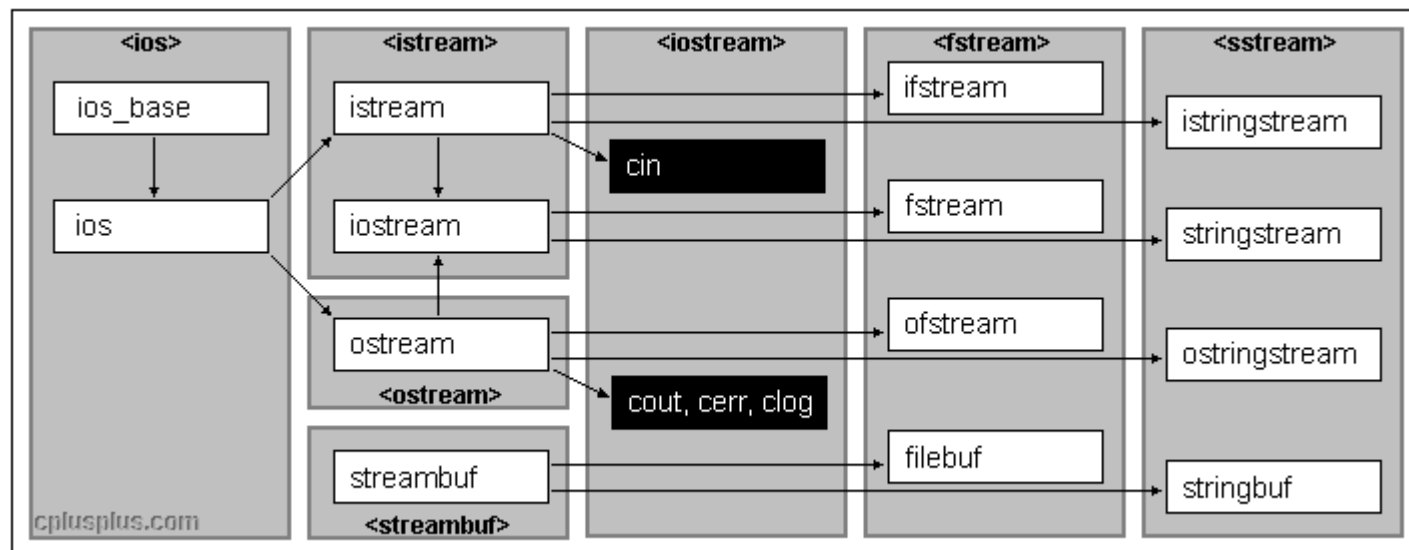
ostrstream строковые потоки вывода

strstream двунаправленные строковые потоки

ifstream файловые потоки ввода

ofstream файловые потоки вывода

fstream двунаправленные файловые потоки



Поточный ввод-вывод

Библиотека `iostream` определяет три стандартных потока:

- ❑ `cin` стандартный входной поток
- ❑ `cout` стандартный выходной поток
- ❑ `cerr` стандартный поток вывода сообщений об ошибках

Для их использования в Microsoft Visual Studio необходимо прописать строку:

```
using namespace std;
```

Для выполнения операций ввода-вывода переопределены две операции поразрядного сдвига:

- ❑ `>>` получить из входного потока
- ❑ `<<` поместить в выходной поток

Вывод информации

Вывод информации

```
cout << значение;
```

Здесь значение преобразуется в последовательность символов и выводится в выходной поток:

```
cout << n;
```

Возможно многократное назначение потоков:

```
cout << 'значение1' << 'значение2' << ... << 'значение n';
```

```
int n;  
char j;  
cin >> n >> j;  
cout << "Значение n равно" << n << "j=" << j;
```

Вывод информации

Возможно многократное назначение потоков:

```
cout << 'значение1' << 'значение2' << ... << 'значение n';
```

```
int n;  
char j;  
cin >> n >> j;  
cout << "Значение n равно" << n << "j=" << j;
```

Ввод информации

При этом из входного потока читается последовательность символов до пробела, затем эта последовательность преобразуется к типу идентификатора, и получаемое значение помещается в **идентификатор**:

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

Возможно многократное назначение потоков:

```
cin >> переменная1 >> переменная2 >>...>> переменнаяn;
```

При наборе данных на клавиатуре значения для такого оператора должны быть разделены символами (пробел, `\n`, `\t`).


```
int n;  
char j;  
cin >> n >> j;
```

Ввод информации

`basic_istream::operator>>`

Вызывает функцию для входного потока или считывает форматированные данные из входного потока.

C++

 Копировать

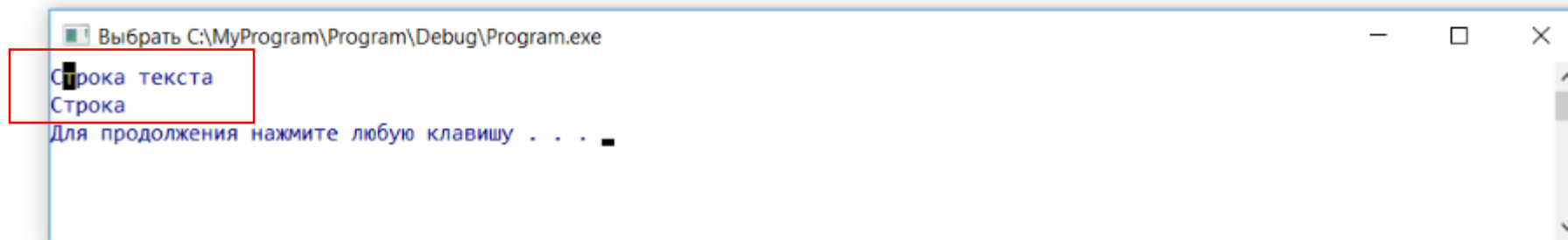
```
basic_istream& operator>>(basic_istream& (* Pfn)(basic_istream&));
basic_istream& operator>>(ios_base& (* Pfn)(ios_base&));
basic_istream& operator>>(basic_ios<Char_T, Tr>& (* Pfn)(basic_ios<Char_T, Tr>&));
basic_istream& operator>>(basic_streambuf<Char_T, Tr>* strbuf);
basic_istream& operator>>(bool& val);
basic_istream& operator>>(short& val);
basic_istream& operator>>(unsigned short& val);
basic_istream& operator>>(int& val);
basic_istream& operator>>(unsigned int& val);
basic_istream& operator>>(long& val);
basic_istream& operator>>(unsigned long& val);
basic_istream& operator>>(long long& val);
basic_istream& operator>>(unsigned long long& val);
basic_istream& operator>>(void *& val);
basic_istream& operator>>(float& val);
basic_istream& operator>>(double& val);
basic_istream& operator>>(long double& val);
```

Ввод информации

Пример

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      char s[80];
6      cin >> s;
7      cout << s << endl;
8      system("pause");
9      return 0;
10 }
```

Результат выполнения



Ввод информации

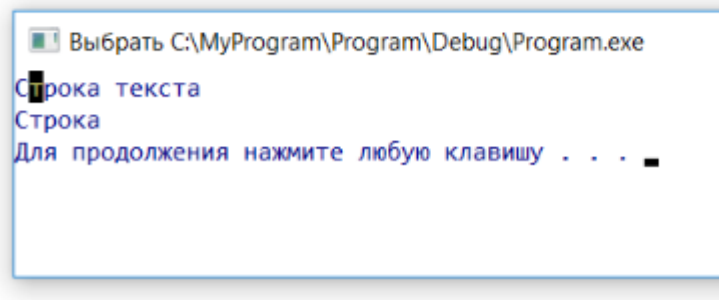
Пример

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      char s[80];
6      cin >> s;
7      cout << s << endl;
8      system("pause");
9      return 0;
10 }
```

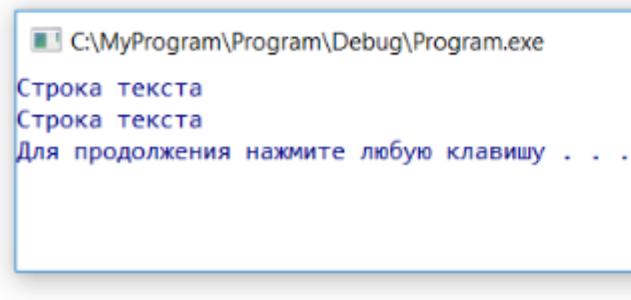
Для ввода текста до символа перевода строки используется манипулятор потока `getline()`:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      char s[80];
6      cin.getline(s, 80);
7      cout << s << endl;
8      system("pause");
9      return 0;
10 }
```

Результат выполнения



Результат выполнения



Ввод информации

```
void prak_0()
{
    system("chcp 1251"); // system("cls");
    const int n = 3;
    char naz[n][15]; char school[n]; int cnt[n];
    for (int i = 0; i < n; i++)
    {
        cout << "Название: "; cin.getline(naz[i],15);
        cin.clear();
        cout << endl<< naz[i] << endl;
    }
}
```

Для ввода текста до символа перевода строки используется манипулятор потока `getline()`:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      char s[80];
6      cin.getline(s, 80);
7      cout << s << endl;
8      system("pause");
9      return 0;
10 }
```

Манипуляторы потока

Функцию - манипулятор потока можно включать в операции помещения в поток и извлечения из потока (<<, >>).

Ввод информации

```
void prak_0()
{
    system("chcp 1251"); // system("cls");
    const int n = 3;
    char naz[n][15]; char school[n]; int cnt[n];
    for (int i = 0; i < n; i++)
    {
        cout << "Название: "; cin.getline(naz[i],15);
        // cin.clear();
        cout << endl<< naz[i] << endl;
    }
}
```

Консоль отладки Microsoft Visual Studio

Текущая кодовая страница: 1251
Название: 12345678901234567890

12345678901234

Название:

Название:

```
void prak_0()
{
    system("chcp 1251"); // system("cls");
    const int n = 3;
    char naz[n][15]; char school[n]; int cnt[n];
    for (int i = 0; i < n; i++)
    {
        cout << "Название: "; cin.getline(naz[i],15);
        cin.clear();
        cout << endl<< naz[i] << endl;
    }
}
```

Консоль отладки Microsoft Visual Studio

Текущая кодовая страница: 1251
Название: 123456789

123456789

Название: 12345678901234567890

12345678901234

Название:

567890

Ввод информации

```
void prak_0()
{
    system("chcp 1251"); // system("cls");
    const int n = 3;
    char naz[n][15]; char school[n]; int cnt[n];
    for (int i = 0; i < n; i++)
    {
        cout << "Название: "; cin.getline(naz[i],15);
        if (!cin.good()) // Проверяем успешность ввода
        {
            cout << " Ай! " << endl;
            cin.clear(); // сбрасываем флаг ошибки
            // Очищаем буфер
            int ch = 0;
            while ((ch = std::cin.get()) != '\n' && ch != EOF);
        }
        cout << endl << naz[i] << endl;
    }
}
```

C:\Users\Filat\source\repos\Struct-1\Debug\Struct-1.exe

Текущая кодовая страница: 1251
Название: 12345678901234567890
Ай!
12345678901234
Название:

Манипуляторы потока

Манипулятор	Описание
endl	Помещение в выходной поток символа конца строки '\n'
dec	Установка основания 10-ой системы счисления
oct	Установка основания 8-ой системы счисления
hex	Установка основания 16-ой системы счисления
setbase	Вывод базовой системы счисления
width(ширина)	Устанавливает ширину поля вывода
fill('символ')	Заполняет пустые знакоместа значением символа
precision(точность)	Устанавливает количество значащих цифр в числе (или после запятой) в зависимости от использования fixed
fixed	Показывает, что установленная точность относится к количеству знаков после запятой
showpos	Показывает знак + для положительных чисел
scientific	Выводит число в экспоненциальной форме
get()	Ожидает ввода символа
getline(указатель, количество)	Ожидает ввода строки символов. Максимальное количество символов ограничено полем количество

Пример форматированного вывода

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      double a = -112.234;
6      double b = 4.3981;
7      int c = 18;
8      cout << endl << "double number:" << endl;
9      cout << "width(10)" << endl;
10     cout.width(10);
11     cout << a << endl << b << endl;
12     cout << "fill('0')" << endl;
13     cout.fill('0');
14     cout.width(10);
15     cout << a << endl << b << endl;
16     cout.precision(5);
17     cout << "precision(5)" << endl << a << endl << b << endl;
18     cout << "fixed" << endl << fixed << a << endl << b << endl;
19     cout << "showpos" << endl << showpos << a << endl << b << endl;
20     cout << "scientific" << endl << scientific << a << endl << b << endl;
21     cout << endl << "int number:" << endl;
22     cout << showbase << hex << c << " " << showbase << oct << c << " ";
23     cout << showbase << dec << c << endl;
24     cin.get();
25     return 0;
26 }
```

```
double number:
width(10)
-112.234
4.3981
```

```
fill('0')
00-112.234
4.3981
```

```
precision(5)
-112.23
4.3981
fixed
-112.23400
4.39810
```

```
showpos
-112.23400
+4.39810
scientific
-1.12234e+02
+4.39810e+00
```

```
int number:
0x12 022 +18
```

Варианты заданий

Разработать программу, которая вводит фактические данные из таблицы, представленной в Вашем варианте индивидуального задания и выводит на экран таблицу, подобную той, которая находится в индивидуальном задании (включая заголовок и примечания).

Вариант 00

Буддийские монастыри Японии периода Нара			
Название	Школа	Количество монахов	Площадь земли (га)
Тодайдзи	Т	220	368.8
Якусидзи	С	50	54.7
Дайандзи	Д	10	12.2
Примечание: Т - Тэндай; С - Сингон; Д – Дзедзицу			

Варианты заданий

Вариант 00

Буддийские монастыри Японии периода Нара

Название	Школа	Количество монахов	Площадь земли (га)
Тодайдзи	Т	220	368.8
Якусидзи	С	50	54.7
Дайандзи	Д	10	12.2

Примечание: Т - Тэндай; С - Сингон; Д – Дзедзицу

```
cout << endl << endl;
cout << "|"; cout.width(60); cout.fill('-'); cout<<'<endl;
cout.fill(' '); cout.width(60); cout <<left<< "|Буддийские монастыри Японии периода Нара"; cout << '|' << endl;
cout.width(60); cout.fill('-'); cout << '|'; cout << "|" << endl;
cout.fill(' ');
```

```
cout.width(15); cout << "| Название";
cout.width(10); cout << "| Школа";
cout.width(15); cout << "| Количество";
cout.width(20); cout << "| Площадь земли";
cout << '|' << endl;
```

```
cout.width(15); cout << "|";
cout.width(10); cout << "|";
cout.width(15); cout << "| монахов";
```

```
// cout.width(20); cout << "| (га)";
cout << '|'; myCentr("(га)", 20);
cout << '|' << endl;
```

```
cout << "|"; cout.width(60); cout.fill('-'); cout << '|' << endl;
```

Буддийские монастыри Японии периода Нара

Название	Школа	Количество монахов	Площадь земли (га)
----------	-------	--------------------	--------------------

Варианты заданий

Вариант 00

Буддийские монастыри Японии периода Нара			
Название	Школа	Количество монахов	Площадь земли (га)
Тодайдзи	Т	220	368.8
Якусидзи	С	50	54.7
Дайандзи	Д	10	12.2
Примечание: Т - Тэндай; С - Сингон; Д – Дзедзицу			

```
cout << endl << endl;
cout << "|"; cout.width(60); cout.fill('-'); cout<<"|"<<endl;
cout.fill(' '); cout.width(60); cout <<left<< "|Буддийские монастыри Японии периода Нара"; cout << '|' << endl;
cout.width(60); cout.fill('-'); cout << '|'; cout << "|" << endl;
cout.fill(' ');
```

```
cout.width(15); cout << "| Название";
cout.width(10); cout << "| Школа";
cout.width(15); cout << "| Количество";
cout.width(20); cout << "| Площадь земли";
cout << '|' << endl;
```

```
cout.width(15); cout << "|";
cout.width(10); cout << "|";
cout.width(15); cout << "| монахов";
```

```
// cout.width(20); cout << "| (га)";
cout << '|'; myCentr("(га)", 20);
cout << '|' << endl;
```

```
cout << "|"; cout.width(60); cout.fill('-'); cout << '|' << endl;
```

```
// void myCentr(const char s[], int wLine)
void myCentr(const char* s , int wLine)
{
    int w = strlen(s);
    int delta = (wLine - w) / 2 -1;
    cout << left;
    cout.width(delta); cout << " ";
    cout<< s;
    cout.width(delta); cout << " " ;
    // if (...)
    // ...
}
```

<ios>

Independent flags (switch on):

<u>boolalpha</u>	Alphanumerical bool values <small>(function)</small>
<u>showbase</u>	Show numerical base prefixes <small>(function)</small>
<u>showpoint</u>	Show decimal point <small>(function)</small>
<u>showpos</u>	Show positive signs <small>(function)</small>
<u>skipws</u>	Skip whitespaces <small>(function)</small>
<u>unitbuf</u>	Flush buffer after insertions <small>(function)</small>
<u>uppercase</u>	Generate upper-case letters <small>(function)</small>

Independent flags (switch off):

<u>noboolalpha</u>	No alphanumerical bool values <small>(function)</small>
<u>noshowbase</u>	Do not show numerical base prefixes <small>(function)</small>
<u>noshowpoint</u>	Do not show decimal point <small>(function)</small>
<u>noshowpos</u>	Do not show positive signs <small>(function)</small>
<u>noskipws</u>	Do not skip whitespaces <small>(function)</small>
<u>nounitbuf</u>	Do not force flushes after insertions <small>(function)</small>
<u>nouppercase</u>	Do not generate upper case letters <small>(function)</small>

Numerical base format flags ("basefield" flags):

<u>dec</u>	Use decimal base <small>(function)</small>
<u>hex</u>	Use hexadecimal base <small>(function)</small>
<u>oct</u>	Use octal base <small>(function)</small>

Floating-point format flags ("floatfield" flags):

<u>fixed</u>	Use fixed floating-point notation <small>(function)</small>
<u>scientific</u>	Use scientific floating-point notation <small>(function)</small>

Adjustment format flags ("adjustfield" flags):

<u>internal</u>	Adjust field by inserting characters at an internal position <small>(function)</small>
<u>left</u>	Adjust output to the left <small>(function)</small>
<u>right</u>	Adjust output to the right <small>(function)</small>

