



§ 15. 输入输出流

1. C/C++的标准输入输出流及C++的文件操作(已讲过, 请复习相关课件)

相关内容:

第03模块:

3. 4. C++的输入与输出

注: 含C方式的格式化输入与输出

第08模块: 输入输出流

8. 1. C++的输入与输出

8. 2. 标准输入流

8. 3. 标准输出流

8. 4. 文件操作与文件流

补充:

★ cin提取数据后, 会根据数据类型是否符合要求而返回逻辑值

● 当cin返回为1/true时, 读入的值才可信

=>正确的处理逻辑: cin读入后, 先判断cin, 为1再取值

● 不同编译器, cin为0时, a的值可能不同(不可信)

● 还可以用cin.good()/cin.fail()来判断

注意: cin.good()与cin.fail()不是任何时候都互斥的!

右例: 观察什么时候in.good()和in.fail()同时为0!!!

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    ifstream in("a.txt", ios::in);
    char buf[128];
    if (in.is_open() == 0) {
        cout << "无法打开文件" << endl;
        return -1;
    }
    while (1) {
        in.getline(buf, sizeof(buf));
        cout << bool(in) << ' ' << in.good() << ' ' << in.fail() << endl;
        if (in.eof())
            break;
        cout << '*' << buf << '*' << endl;
    }
    in.close();
    return 0;
}
```

另: 建立a.txt文件, 两行

1234567 张三

7654321 李四

注: 如果是记事本编辑, 记得编码为ANSI

分两种情况运行:

最后一行后有换行/无换行



§ 15. 输入输出流

2. C语言的文件操作

2.1. 文件指针

FILE *文件指针变量

- ★ FILE是系统定义的结构体
- ★ C语言中文件操作的基本依据，所有针对文件的操作均要依据该指针
- ★ #include <stdio.h> (VS可以不需要)
- ★ VS以为不安全，需要加 #define _CRT_SECURE_NO_WARNINGS
- ★ 文件读写后，文件指针会自动后移



§ 15. 输入输出流

2. C语言的文件操作

2.2. 文件的打开与关闭

假设: `FILE *fp`定义一个文件指针

2.2.1. 文件的打开

`FILE *fopen(文件名, 打开方式)`

`fp = fopen("test.dat", "r");`

`fp = fopen("c:\\demo\\test.dat", "w");` //也可表示为: `"c:/demo/test.dat"`

★ 打开的基本方式如下:

r: 只读方式

w: 只写方式

a: 追加方式

+: 可读可写

b: 二进制

t: 文本方式(缺省)

★ 打开的基本方式及组合见右表

★ 若带路径, 则\必须用\\表示(也可以/)

★ 若打开不成功, 则返回NUL

2.2.2. 文件的关闭

`fclose(文件指针)`

`fclose(fp);`

打开方式	意义
r/rt	只读方式打开文本文件(不存在则失败)
w/wt	只写方式打开或建立文本文件(存在则清零)
a/at	追加写方式打开或建立文本文件(头读尾写)
rb	只读方式打开二进制文件(不存在则失败)
wb	只写打开或建立二进制文件(存在则清零)
ab	追加写方式打开或建立二进制文件(头读尾写)
r+/rt+	读写方式打开文本文件(不存在则失败)
w+/wt+	读写方式创建文本文件(存在则清零)
a+/at+	读+追加写方式打开或建立文本文件(头读尾写)
rb+	读写方式打开二进制文件(不存在则失败)
wb+	读写方式创建二进制文件(存在则清零)
ab+	读+追加写方式打开二进制文件(头读尾写)



§ 15. 输入输出流

2. C语言的文件操作

2.3. 文本文件的读写

2.3.1. 按字符读写文件

读: `int fgetc(文件指针)`

- 返回读到字符的ASCII码 (返回值同`getchar`)

写: `int fputc(字符常量/变量, 文件指针)`

- 返回写入字符的ASCII码 (返回值同`putchar`)

★ 必须保证文件的打开方式符合要求

```
char ch1;  
ch1=fgetc(fp);
```

```
char ch2 = 'A';  
fputc(ch2, fp);
```

2.3.2. 判断文件是否到达尾部

`int feof(文件指针)`

★ 若到达尾部, 返回1, 否则为0

2.3.3. 按格式读写文件

读: `int fscanf(文件指针, 格式串, 输入表列)`

- 返回读取正确的数量 (返回值同`scanf`)

写: `int fprintf(文件指针, 格式串, 输出表列)`

- 返回输出字符的个数 (返回值同`printf`)

★ 格式串、输入/输出表列的使用同`scanf/printf`

```
int i;  
char ch;  
fscanf(fp, "%d%c", &i, &ch);
```

```
int i=10;  
char ch='A';  
fprintf(fp, "%d%c", i, ch);
```



§ 15. 输入输出流

2. C语言的文件操作

2.3. 文本文件的读写

2.3.4. 用文件方式进行标准输入输出

stdin : 标准输入设备

stdout : 标准输出设备

stderr : 错误输出设备

} 这三个是系统预置的FILE *, 直接用, 不需要打开关闭

```
#define _CRT_SECURE_NO_WARNINGS
```

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

```
int main()
```

```
{
```

```
    int i;
```

```
    char ch;
```

```
    ch = fgetc(stdin);
```

等价
⇔ getchar();

```
    putchar(ch);
```

```
    fputc('A', stdout);
```

⇔ putchar('A');

```
    fscanf(stdin, "%d", &i);
```

⇔ scanf("%d", &i);

```
    fprintf(stdout, "i=%d\n", i);
```

⇔ printf("i=%d\n", i);

```
    fprintf(stderr, "i=%d\n", i);
```

⇔ cerr<<"i="<<i<< endl;

//C方式无专用错误输出

//perror() 功能不同

```
    return 0;
```

```
}
```



§ 15. 输入输出流

2. C语言的文件操作

2.3. 文本文件的读写

2.3.5. 用freopen重定向标准输入输出

★ FILE *freopen(文件名, 打开方式, 原FILE *);

功能: 将已存在的FILE *映射为另一个新的FILE *

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fp;
    if ((fp = freopen("out.txt", "w", stdout)) == NULL) {
        printf("freopen failed!\n");
        return -1;
    }
    printf("Hello, world!\n");
    fclose(fp);

    return 0;
}
```

- 1、正常运行, 观察运行结果
- 2、不删除已存在的out.txt, 换成"r", 观察运行结果
- 3、先删除已存在的out.txt, 换成"r", 观察运行结果
- 4、如果在fclose的后面再加printf, 能否正常输出?
如果可以, 输出到哪里了? 如果没有, 为什么?

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fp;
    int a, b;
    if ((fp = freopen("in.txt", "r", stdin)) == NULL) {
        printf("freopen failed!\n");
        return -1;
    }
    scanf("%d %d", &a, &b);
    printf("a=%d b=%d\n", a, b);
    fclose(fp);

    return 0;
}
```

- 1、在当前目录下建立in.txt文件, 写入两个整数, 观察运行结果
- 2、在当前目录下没有/删除in.txt的情况下运行, 观察运行结果
- 3、如果在fclose的后面再加scanf, 能否正常输入?
如果可以, 从哪里读? 如果不行, 为什么?



§ 15. 输入输出流

2. C语言的文件操作

2.3. 文本文件的读写

2.3.5. 用freopen重定向标准输入输出

- ★ FILE *freopen(文件名, 打开方式, 原FILE *);
功能: 将已存在的FILE *映射为另一个新的FILE *
- ★ 用freopen可以重定向普通文件 (一般不用)

观察运行结果

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    FILE *fp, *fdup;
    if ((fp = fopen("out.txt", "w")) == NULL) {
        printf("fopen out.txt failed!\n");
        return -1;
    }
    fprintf(fp, "Hello, world!\n");

    if ((fdup = freopen("out_dup.txt", "w", fp)) == NULL) {
        printf("freopen failed!\n");
        fclose(fp);
        return -1;
    }
    fprintf(fp, "I am a student.\n"); //注意: 不是fdup !!!

    fclose(fp);
    fclose(fdup);
    return 0;
}
```



§ 15. 输入输出流

2. C语言的文件操作

2.3. 文本文件的读写

2.3.6. 用popen/pclose与系统命令进行交互

- ★ VS下是_popen与_pclose
- ★ Linux下的popen与pclose
- ★ Dev C++下popen/pclose/_popen/_pclose均可

★ Windows示例(分两步操作)

```
//假设编译为 D:\VS-Demo\Debug\demo-cpp.exe
#include <iostream>
using namespace std;
int main()
{
    cout << "Welcome to Tongji University!" << endl;
    return 0;
}

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    FILE* fp = _popen("D:\\VS-Demo\\Debug\\demo-cpp.exe", "r");
    if (fp == NULL) {
        printf("popen failed!\n");
        return -1;
    }
    char ch;
    while (1) {
        ch = fgetc(fp);
        if (feof(fp));
        break;
        putchar(ch);
    }
    _pclose(fp);
    return 0;
}
```

Step1

Step2

再换为 "dir C:\Windows"

★ Linux示例(分两步操作)

```
//假设编译为 /home/u1234567/test
#include <iostream>
using namespace std;
int main()
{
    cout << "Welcome to Tongji University!" << endl;
    return 0;
}

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int main()
{
    FILE* fp;
    if ((fp = popen("/home/u1234567/test", "r")) == NULL) {
        printf("popen failed!\n");
        return -1;
    }
    char ch;
    while (1) {
        ch = fgetc(fp);
        if (feof(fp))
            break;
        putchar(ch);
    }
    pclose(fp);
    return 0;
}
```

Step1

Step2

再换为 "ls -l /etc"



§ 15. 输入输出流

2. C语言的文件操作

2.4. 二进制文件的读写

2.4.1. 按字符读写文件

读: `int fgetc(文件指针)`

- 返回读到字符的ASCII码 (返回值同`getchar`)

写: `int fputc(字符常量/变量, 文件指针)`

- 返回写入字符的ASCII码 (返回值同`putchar`)

★ 必须保证文件的打开方式符合要求

★ 同C++方式, 仅能按字符读写, 且文件中不能有0x1A

2.4.2. 按块读写文件

读: `int fread(缓冲区首址, 块大小, 块数, 文件指针)`

- ★ 返回读满的块数

写: `int fwrite(缓冲区首址, 块大小, 块数, 文件指针)`

- ★ 返回写入成功的块数



§ 15. 输入输出流

2. C语言的文件操作

2.5. 文件指针的移动

2.5.1. 指针复位(回到开头)

rewind(文件指针)

例: rewind(fp);

2.5.2. 任意移动

fseek(文件指针, 位移量, 位移方式)

例: fseek(fp, 123, SEEK_SET): } 从开始移动
fseek(fp, 78, SEEK_CUR): } 从当前位置移动
fseek(fp, -25, SEEK_CUR):
fseek(fp, -57, SEEK_END): 从最后移动

- ★ SEEK_SET的位移必须为正
- SEEK_CUR的位移可正可负
- SEEK_END的位移必须为负

2.5.3. 求文件指针的当前位置

long ftell(文件指针)

例: ftell(fp);

- ★ 从开始位置计算



§ 15. 输入输出流

3. C++的字符串流 (sstream)

3.1. 基本概念

以内存中的string类型变量为输入/输出对象

★ 可以存放各种类型的数据

★ 与标准输入输出流相同，进行文本和二进制之间的相互转换

向string存数据 ⇔ cout: 二进制 => ASCII

从string取数据 ⇔ cin : ASCII => 二进制

● 推论：可用于不同数据类型的转换

★ 不是文件，不需要打开和关闭

3.2. 相关流对象的建立

字符串输出流对象：

ostreamstream 对象名

字符串输入流对象：

istreamstream 对象名

字符串输入/输出流对象：

stringstream 对象名

★ 加 `#include <sstream>`



§ 15. 输入输出流

3. C++的字符串流 (sstream)

3.3. 字符串输出流对象的使用

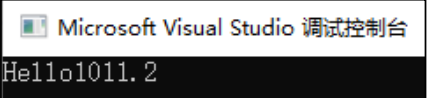
例1: 观察cout的输出

```
#include <iostream>
#include <sstream>
using namespace std;

int main()
{
    ostringstream out;
    out << "Hello" << 10 << 11.2 << endl;

    string s1 = out.str();
    cout << s1 << endl;

    return 0;
}
```



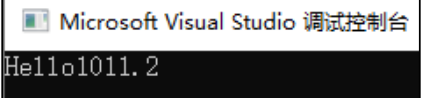
例2: 观察cout的输出

```
#include <iostream>
#include <sstream>
using namespace std;

int main()
{
    ostringstream out;
    out << "Hello" << 10 << 11.2 << endl;

    cout << out.str() << endl; //等价例1

    return 0;
}
```



★ 成员函数str()的作用: 将ostringstream的内容转换为string格式

★ ostringstream最简单的用法: 将多个格式化内容拼在一起, 集中输出



§ 15. 输入输出流

3. C++的字符串流 (sstream)

3.4. 字符串输入流对象的使用

```
#include <iostream>
#include <sstream>
using namespace std;
```

```
int main()
{
```

```
    istringstream in("Hello 10 11.2");
    cout << in.str() << endl;
```

```
    char s[10];
    short i;
    float f;
    in >> s >> i >> f;
    cout << s << '-' << i << '-' << f << endl;
```

```
    cout << in.good() << endl;
    cout << in.str() << endl;
```

```
    return 0;
```

```
}
```

例1: 观察cout的输出

```
Microsoft Visual Studio 调试控制台
Hello 10 11.2
Hello-10-11.2
0
Hello 10 11.2
```

- ★ 可用str()打印现有内容
- ★ 读完后, 内容仍在
- ★ 如果现有内容全部读完, goodbit会置0

```
#include <iostream>
#include <sstream>
using namespace std;
```

```
int main()
{
```

```
    istringstream in("Hello 10 11.2 xyz");
    cout << in.str() << endl;
```

```
    char s[10];
    short i;
    float f;
    in >> s >> i >> f;
    cout << s << '-' << i << '-' << f << endl;
```

```
    cout << in.good() << endl;
    cout << in.str() << endl;
```

```
    return 0;
```

```
}
```

例2: 观察cout的输出

```
Microsoft Visual Studio 调试控制台
Hello 10 11.2 xyz
Hello-10-11.2
1
Hello 10 11.2 xyz
```



§ 15. 输入输出流

3. C++的字符串流(sstream)

3.4. 字符串输入流对象的使用

```
#include <iostream>
#include <sstream>
using namespace std;
```

例3: 观察cout的输出

```
int main()
{
    istringstream in("Hello 10 11.2");
    cout << in.str() << endl;

    char s1[10], s2[10]="xyz";
    short i1, i2=123;
    float f1, f2=0.456F;
    in >> s1 >> i1 >> f1;
    cout << s1 << '-' << i1 << '-' << f1 << endl;
    cout << s2 << '-' << i2 << '-' << f2 << endl;

    in.clear();
    in.seekg(0, ios::beg);
    in >> s2 >> i2 >> f2;
    cout << s2 << '-' << i2 << '-' << f2 << endl;
    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
Hello 10 11.2
Hello-10-11.2
xyz-123-0.456
Hello-10-11.2
```

★ istringstream的内容可重复读取



§ 15. 输入输出流

3. C++的字符串流(ssstream)

3. 4. 字符串输入流对象的使用

```
#include <iostream>
#include <sstream>
using namespace std;
```

例4: 观察cout的输出

```
int main()
{
    istringstream in("Hello 70000 11.2");

    char s[10];
    short i;
    float f;

    in >> s;

    in >> i;

    in >> f;
    cout << s << '-' << i << '-' << f << endl;

    return 0;
}
```

Microsoft Visual Studio 调试控制台
Hello-32767--1.07374e+08

```
#include <iostream>
#include <sstream>
using namespace std;
```

例5: 观察cout的输出

```
int main()
{
    istringstream in("Hello 70000 11.2");

    char s[10];
    short i;
    float f;

    in >> s;
    cout << in.good() << endl;
    in >> i;
    cout << in.good() << endl;

    in >> f;
    cout << s << '-' << i << '-' << f << endl;

    return 0;
}
```

Microsoft Visual Studio 调试控制台
1
0
Hello-32767--1.07374e+08

```
#include <iostream>
#include <sstream>
using namespace std;
```

例6: 观察cout的输出

```
int main()
{
    istringstream in("Hello 70000 11.2");

    char s[10];
    short i;
    float f;

    in >> s;
    cout << in.good() << endl;
    in >> i;
    cout << in.good() << endl;
    in.clear();
    in >> f;
    cout << s << '-' << i << '-' << f << endl;

    return 0;
}
```

Microsoft Visual Studio 调试控制台
1
0
Hello-32767-11.2

★ 如果数据超范围, 后续会错, 可clear()恢复



§ 15. 输入输出流

3. C++的字符串流 (sstream)

3.4. 字符串输入流对象的使用

```
#include <iostream>
#include <sstream>
using namespace std;
```

例7: 观察cout的输出

```
int main()
{
    istringstream in("Hello 10 11.2");

    char s[10];
    short i;
    float f;
    in >> s >> i >> f;
    cout << s << '-' << i << '-' << f << endl;

    cout << in.good() << endl;
    in.str("tongji 123 0.123");

    in >> s >> i >> f;
    cout << s << '=' << i << '=' << f << endl;

    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
Hello-10-11.2
0
=10=11.2
```

```
#include <iostream>
#include <sstream>
using namespace std;
```

例8: 观察cout的输出

```
int main()
{
    istringstream in("Hello 10 11.2 12345");

    char s[10];
    short i;
    float f;
    in >> s >> i >> f;
    cout << s << '-' << i << '-' << f << endl;

    cout << in.good() << endl;
    in.str("tongji 123 0.123");

    in >> s >> i >> f;
    cout << s << '=' << i << '=' << f << endl;

    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
Hello-10-11.2
1
tongji=123=0.123
```

★ 可用带参str()再次赋新内容，但注意goodbit



§ 15. 输入输出流

3. C++的字符串流 (sstream)

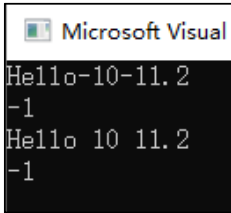
3.5. 字符串输入/输出流对象的使用

例1: 观察cout的输出

```
#include <iostream>
#include <sstream>
using namespace std;
int main()
{
    stringstream ss("Hello 10 11.2");
    char s[10];
    short i;
    float f;
    ss >> s >> i >> f;
    cout << s << '-' << i << '-' << f << endl;
    cout << ss.tellg() << endl;

    ss << "xyz 123 0.456";
    cout << ss.str() << endl;
    cout << ss.tellg() << endl;

    return 0;
}
```

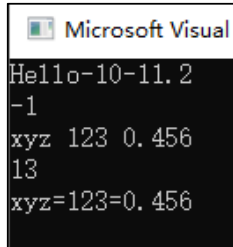


例2: 观察cout的输出

```
#include <iostream>
#include <sstream>
using namespace std;
int main()
{
    stringstream ss("Hello 10 11.2");
    char s[10];
    short i;
    float f;
    ss >> s >> i >> f;
    cout << s << '-' << i << '-' << f << endl;
    cout << ss.tellg() << endl;

    ss.clear();
    ss << "xyz 123 0.456";
    cout << ss.str() << endl;
    cout << ss.tellg() << endl;

    ss.seekg(0, ios::beg);
    ss >> s >> i >> f;
    cout << s << '=' << i << '=' << f << endl;
    return 0;
}
```



★ stringstream可读可写，但注意goodbit



§ 15. 输入输出流

3. C++的字符串流(ssstream)

3. 5. 字符串输入/输出流对象的使用

//先从流对象中输入数据，再把排序后的结果输出到流对象中

例3：综合应用

```
#include <iostream>
#include <sstream>
using namespace std;
#define N 10
int main()
{
    stringstream ss("12 34 65 -23 -32 33 61 99 321 32");
    int a[N], i, j, t;

    for (i = 0; i < N; i++)
        ss >> a[i]; //ss中的内容逐个读入int a[10]中

    cout << "array a:";
    for (i = 0; i < N; i++) //输出int a[10]的内容
        cout << a[i] << " ";
    cout << endl;
    //进行排序
    for (i = 0; i < N - 1; i++)
        for (j = 0; j < N - 1 - i; j++)
            if (a[j] > a[j + 1]) {
                t = a[j];
                a[j] = a[j + 1];
                a[j + 1] = t;
            }

    //输出到ss中(ss刚才用做了输入流，现在覆盖其中的内容)
    ss.clear();
    ss.seekg(0, ios::beg);
    for (i = 0; i < N; i++)
        ss << a[i] << " ";
    ss << endl;
    cout << "array a after sort:" << ss.str() << endl;

    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
array a:12 34 65 -23 -32 33 61 99 321 32
array a after sort:-32 -23 12 32 33 34 61 65 99 321
```



§ 15. 输入输出流

3. C++的字符串流(ssstream)

3.1. 基本概念

3.2. 相关流对象的建立

3.3. 字符串输出流对象的使用

3.4. 字符串输入流对象的使用

3.5. 字符串输入/输出流对象的使用

总结:

★ 存储形式为string, 不需要用户考虑空间

★ 使用方式同iostream/fstream基本相似(部分细节可能不同)

★ 如果结果与预期不同, 多判断good()/fail()

★ C++还有一个stringstream系列, 但是在新标准中已是deprecated

● 要求: 能读懂别人用stringstream写的代码, 自己不准用!!!



§ 15. 输入输出流

3. C++的字符串流 (sstream)

3.6. 用字符串流对象实现不同数据类型的转换

```
#include <iostream>
#include <sstream>
using namespace std;
```

例1: 字符串转double

```
int main()
{
    istringstream in("123.456");
    double d;

    in >> d;
    cout << d << endl;

    return 0;
}
```

```
#include <iostream>
#include <sstream>
using namespace std;
```

例2: double转字符串

```
int main()
{
    ostringstream out;
    double d = 123.456;
    char str[10];

    out << d;
    strcpy(str, out.str().c_str());
    cout << str << endl;

    return 0;
}
```

```
#include <iostream>
#include <sstream>
using namespace std;
string tj_to_string(const double d)
```

例3: 多种类型转字符串
(重载方式)

```
{
    ostringstream out;
    out << d;
    return out.str();
}
string tj_to_string(const int i)
{
    ostringstream out;
    out << i;
    return out.str();
}
string tj_to_string(const char ch)
{
    ostringstream out;
    out << ch;
    return out.str();
}
int main()
{
    string s1 = tj_to_string(123.456);
    string s2 = tj_to_string(12345);
    string s3 = tj_to_string('A');

    cout << s1 << endl;
    cout << s2 << endl;
    cout << s3 << endl;
}
```



§ 15. 输入输出流

4. C语言中实现与C++的字符串流相似的功能

4.1. 向字符数组输出格式化的数据

int sprintf(字符数组, "格式串", 输出表列);

★ 返回值是输出字符的个数

指不同类型数据按格式串的要求转换为文本方式后字符的个数

★ 与printf相同, 完成二进制向ASCII的转换

★ VS下需加 #define _CRT_SECURE_NO_WARNINGS

```
//例: 将不同数据输出到ostringstream中
#include <iostream>
#include <sstream>
using namespace std;

int main()
{
    ostringstream out;

    out << "Hello" << 10 << 11.2 << endl;
    cout << out.str() << endl;

    return 0;
}
```

Microsoft Visual Studio 调试控制台
Hello1011.2



```
//将不同数据输出到字符数组中
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    char c[80];
    int ret;
    ret = sprintf(c, "%s%d%.1f", "Hello", 10, 11.2);
    printf("%s\n", c);
    printf("ret=%d\n", ret); //搞懂ret的含义

    return 0;
}
```

Microsoft Visual Studio 调试控制台
Hello1011.2
ret=11



§ 15. 输入输出流

4. C语言中实现与C++的字符串流相似的功能

4.1. 向字符数组输出格式化的数据

int sprintf(字符数组, "格式串", 输出表列);

//例: 接结构体的内容输出到一维字符数组中

```
#define _CRT_SECURE_NO_WARNINGS
```

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

```
struct student {
```

```
    int num;
```

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

```
    float score;
```

```
};
```

```
int main()
```

```
{
```

```
    struct student stud[3]={1001,"Li",78, 1002,"Wang",89.5, 1004,"Fun",90};
```

```
    char c[50], *s = c;
```

```
    for (int i=0; i<3; i++)
```

```
        s+=sprintf(s, "%d %s %.1f", stud[i].num , stud[i].name, stud[i].score);
```

```
    printf("array c:%s\n", c);
```

```
    return 0;
```

```
}
```

Microsoft Visual Studio 调试控制台

array c:1001 Li 78.01002 Wang 89.51004 Fun 90.0

多次向字符数组输出格式化数据
(注意: 和C++方式的不同)

想明白s+=sprintf(s, "");的用法



§ 15. 输入输出流

4. C语言中实现与C++的字符串流相似的功能

4.2. 从字符数组中输入格式化的数据

int sscanf(字符数组, "格式串", 输入表列);

- ★ 返回值是正确读入的输入数据的个数
- ★ 与scanf相同, 完成ASCII向二进制的转换
- ★ VS下需加 #define _CRT_SECURE_NO_WARNINGS

```
#include <iostream>
#include <sstream>
using namespace std;

int main()
{
    istringstream in("Hello 10 11.2");
    char s[10];
    int i;
    float f;

    in >> s >> i >> f;

    cout << s << i << f << endl;

    return 0;
}
```

Microsoft Visual Studio 调试控制台
Hello1011.2

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    char c[80] = "Hello 10 11.2";
    char s[10];
    int i, ret;
    float f;

    ret = sscanf(c, "%s%d%f", s, &i, &f);
    printf("%s%d%.1f\n", s, i, f);
    printf("ret=%d\n", ret);

    return 0;
}
```

Microsoft Visual Studio 调试控制台
Hello1011.2
ret=3



§ 15. 输入输出流

4. C语言中实现与C++的字符串流相似的功能

4.3. 同时进行输入输出

```
//从字符串中读入10个数并排序，再把排序结果输出到字符串中
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#define N 10
int main()
{
    char ss[80] = "12 34 65 -23 -32 33 61 99 321 32", *s = ss;
    int a[N], i, j, t;
    //ss中的内容逐个读入int a[10]中
    for (i = 0; i < N; i++) {
        sscanf(s, "%d", &a[i]);
        s = strchr(s, ' ');
        s++; //指向空格后的字符
    }
    printf("array a:");
    for (i = 0; i < N; i++) //输出int a[10]的内容
        printf("%d ", a[i]);
    printf("\n");
    //进行排序
    for (i = 0; i < N - 1; i++)
        for (j = 0; j < N - 1 - i; j++)
            if (a[j] > a[j + 1]) {
                t = a[j];
                a[j] = a[j + 1];
                a[j + 1] = t;
            }
    //输出到ss中（ss刚才用做了输入流）
    s = ss; //重新指向ss[0]
    for (i = 0; i < N; i++)
        s+=sprintf(s, "%d ", a[i]);
    s+=sprintf(s, "\n");
    printf("array a after sort:%s\n", ss);
    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
array a:12 34 65 -23 -32 33 61 99 321 32
array a after sort:-32 -23 12 32 33 34 61 65 99 321
```