浙江科技大学
ZHEJIANG UNIVERSITY OF SCIENCE & TECHNOLOGY

《高级语言程序设计》
课程报告

姓名：张梦南

班级：自动化 221

学号：1220310013

自动化与电气工程学院

2023-12

课程设计一：

一、设计要求：

输入一段文本，要求对其中的每个单词进行反序，并按照反序后的单词的首字母排序输出这段文本（首字母相同，按照第二个字母排序，依此类推）。

输出格式与输入格式相同。

二、设计分析：

利用课本第五章（使用顺序容器和分析字符串）中所 split 函数，实现将字符串拆分为单词。参考第六章（使用库算法）中的回文程序，实现对单词进行反转。最后按照反转后单词的首字母对单词进行排序。

三、主要代码分析：

```
1.  string reverseString(const string& s) {
2.      return string(s.rbegin(), s.rend());
3.  }
```

借助 STL 库实现。s.rbegin()是反向迭代器，指向容器的最后一个元素。s.rend()也是反向迭代器，指向第一个元素前面的位置。用了 string 的其中一种构造函数：string(Iter begin, Iter end)。

```
1.    std::sort(words.begin(), words.end());
```

利用 sort 函数来实现根据首字母对单词排序。

四、程序运行示例：

课程设计一程序运行示例如图 1.1 所示。

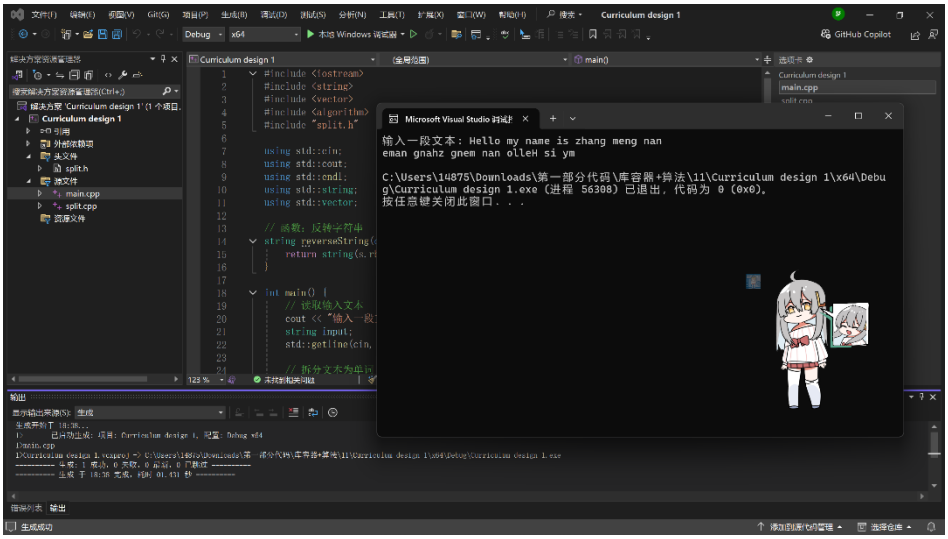

图 1.1 课程设计一程序运行示例

五、附完整代码：

main.cpp：

```cpp
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include "split.h"

using std::cin;
using std::cout;
using std::endl;
using std::string;
using std::vector;

// 函数：反转字符串
string reverseString(const string& s) {
    return string(s.rbegin(), s.rend());
}

int main() {
    // 读取输入文本
    cout << "输入一段文本：";
    string input;
    std::getline(cin, input);

    // 拆分文本为单词
    vector<string> words = split(input);

```

```
    // 对每个单词进行反转
    for (auto& word : words) {
        word = reverseString(word);
    }

    // 按照反转后的单词排序
    std::sort(words.begin(), words.end());

    // 输出排序后的单词，保持与原输入格式一致
    for (size_t i = 0; i < words.size(); ++i) {
        if (i > 0) {
            cout << " "; // 单词间加空格
        }
        cout << words[i];
    }
    cout << endl;

    return 0;
}
```

split.h:

```
#ifndef GUARD_split_h
#define GUARD_split_h

#include <vector>
#include <string>
std::vector<std::string> split(const std::string&);

#endif
```

split.cpp:

```
#include <cctype>
#include <string>
#include <vector>

#include "split.h"

using std::vector;
using std::string;

```

```
#ifndef _MSC_VER
using std::isspace;
#endif

vector<string> split(const string& s)
{
    vector<string> ret;
    typedef string::size_type string_size;
    string_size i = 0;

    // invariant: we have processed characters `['original value of `i', `i)'
    while (i != s.size()) {
        // ignore leading blanks
        // invariant: characters in range `['original `i', current `i)' are all spaces
        while (i != s.size() && isspace(s[i]))
            ++i;

        // find end of next word
        string_size j = i;
        // invariant: none of the characters in range `['original `j', current `j)' is a space
        while (j != s.size() && !isspace(s[j]))
            ++j;

        // if we found some nonwhitespace characters
        if (i != j) {
            // copy from `s' starting at `i' and taking `j' `\-' `i' chars
            ret.push_back(s.substr(i, j - i));
            i = j;
        }

    }
    return ret;
}
```

课程设计二：

一、设计要求：

输入一段文本，将其中的 http 协议的 url 输出，按照其单词首字母次序输出（首字母相同，按照第二个字母排序，依此类推）。

二、设计分析：

利用第六章（使用库算法）中的查找 URL 程序所使用的 urls 函数，来实现从文本中提取出所有有效的 URL。按照 URL 的单词首字母次序进行排序。

三、主要代码分析：

```
1.  vector<string> urls = find_urls(input);
```

利用 urls 函数从输入文本中提取出 URL 链接。

```
1.  vector<string> http_urls;
2.      for (const string& url : urls) {
3.          if (url.find("http://") == 0 || url.find("https://") == 0) {
4.              http_urls.push_back(url);
5.          }
6.      }
```

将提取出来的 URL 以 http 或 https 分成两类（结果优先打印输出 http）

```
1.  sort(http_urls.begin(), http_urls.end());
```

利用 sort 函数来实现根据首字母对单词排序。（与课程设计一中的排序实现方式相同）

四、程序运行示例：

课程设计二程序运行示例如图 2.1 所示。

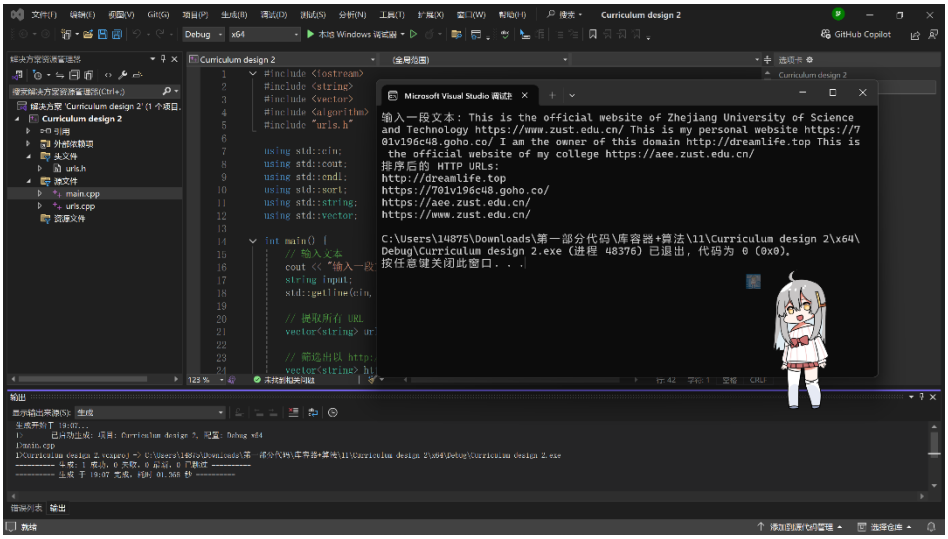

图 2.1 课程设计二程序运行示例

五、附完整代码：

main.cpp：

```cpp
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include "urls.h"

using std::cin;
using std::cout;
using std::endl;
using std::sort;
using std::string;
using std::vector;

int main() {
    // 输入文本
    cout << "输入一段文本: ";
    string input;
    std::getline(cin, input);

    // 提取所有 URL
    vector<string> urls = find_urls(input);

    // 筛选出以 http:// 或 https:// 开头的 URL
    vector<string> http_urls;
    for (const string& url : urls) {
```

```
        if (url.find("http://") == 0 || url.find("https://") == 0) {
            http_urls.push_back(url);
        }
    }

    // 按字母顺序排序
    sort(http_urls.begin(), http_urls.end());

    // 输出结果
    cout << "排序后的 HTTP URLs:" << endl;
    for (const string& url : http_urls) {
        cout << url << endl;
    }

    return 0;
}
```

urls.h:

```
#ifndef GUARD_urls_h
#define GUARD_urls_h

#include <vector>
#include <string>

std::vector<std::string> find_urls(const std::string& s);

#endif
```

urls.cpp:

```
#include <algorithm>
#include <cctype>
#include <string>
#include <vector>

#include "urls.h"

using std::find;
using std::find_if;

```

```
#ifndef _MSC_VER
using std::isalnum;
using std::isalpha;
using std::isdigit;
#endif

using std::search;
using std::string;
using std::vector;

bool not_url_char(char);

string::const_iterator
url_end(string::const_iterator, string::const_iterator);

string::const_iterator
url_beg(string::const_iterator, string::const_iterator);
vector<string> find_urls(const string& s)
{
    vector<string> ret;
    typedef string::const_iterator iter;
    iter b = s.begin(), e = s.end();

    // look through the entire input
    while (b != e) {

        // look for one or more letters followed by `://'
        b = url_beg(b, e);

        // if we found it
        if (b != e) {
            // get the rest of the \s-1URL\s0
            iter after = url_end(b, e);

            // remember the \s-1URL\s0
            ret.push_back(string(b, after));

            // advance `b' and check for more \s-
1URL\s0s on this line
            b = after;
        }
    }
    return ret;
}
```

```
54.
55. string::const_iterator
56. url_end(string::const_iterator b, string::const_iterator e)
57. {
58.     return find_if(b, e, not_url_char);
59. }
60.
61. bool not_url_char(char c)
62. {
63.     // characters, in addition to alphanumerics, that can appear in a
     \s-1URL\s0
64.     static const string url_ch = "~;/?:@=&$-_.+!*'(),";
65.
66.     // see whether `c' can appear in a \s-
    1URL\s0 and return the negative
67.     return !(isalnum(c) ||
68.              find(url_ch.begin(), url_ch.end(), c) != url_ch.end());
69. }
70.
71. string::const_iterator
72. url_beg(string::const_iterator b, string::const_iterator e)
73. {
74.     static const string sep = "://";
75.
76.     typedef string::const_iterator iter;
77.
78.     // `i' marks where the separator was found
79.     iter i = b;
80.
81.     while ((i = search(i, e, sep.begin(), sep.end())) != e) {
82.
83.         // make sure the separator isn't at the beginning or end of t
    he line
84.         if (i != b && i + sep.size() != e) {
85.
86.             // `beg' marks the beginning of the protocol-name
87.             iter beg = i;
88.             while (beg != b && isalpha(beg[-1]))
89.                 --beg;
90.
91.             // is there at least one appropriate character before and
     after the separator?
92.             if (beg != i && !not_url_char(i[sep.size()]))
```

```
                return beg;
        }

        // the separator we found wasn't part of a \s-1URL\s0; advance `i' past this separator
        i += sep.size();
    }
    return e;
  }
```

课程设计三：

一、设计要求：

输入学生姓名、期中成绩、期末成绩和平时成绩，计算他的总评成绩。将不及格学生信息和及格学生信息分开输出，并按照姓名首字母排序。

二、设计分析：

利用第四章（组织程序和数据）中所使用的 student_info 函数和 median 函数来储存所输入的学生数据，利用 grade 函数来计算学生的总评成绩。最后按学生姓名字母顺序分别输出及格和不及格学生的信息。

总评成绩 = 20%*期中成绩 + 40%*期末成绩 + 40%*作业成绩

三、主要代码分析：

```
const double PASS_GRADE = 60.0;
```

设置总评及格成绩为 60 分。

```
vector<Student_info> pass, fail;
    for (const auto& student : students) {
        if (grade(student) >= PASS_GRADE) {
            pass.push_back(student);
        }
        else {
            fail.push_back(student);
        }
    }
```

利用循环（for）加判断（if&else）将学生根据总评成绩分为及格和不及格。

```
std::sort(pass.begin(), pass.end(), compare);
std::sort(fail.begin(), fail.end(), compare);
```

利用 sort 函数来实现根据学生名字首字母来排序。（与课程设计一和二中的排序实现方式相同）

四、程序运行示例：

课程设计二程序运行示例如图 3.1 所示。

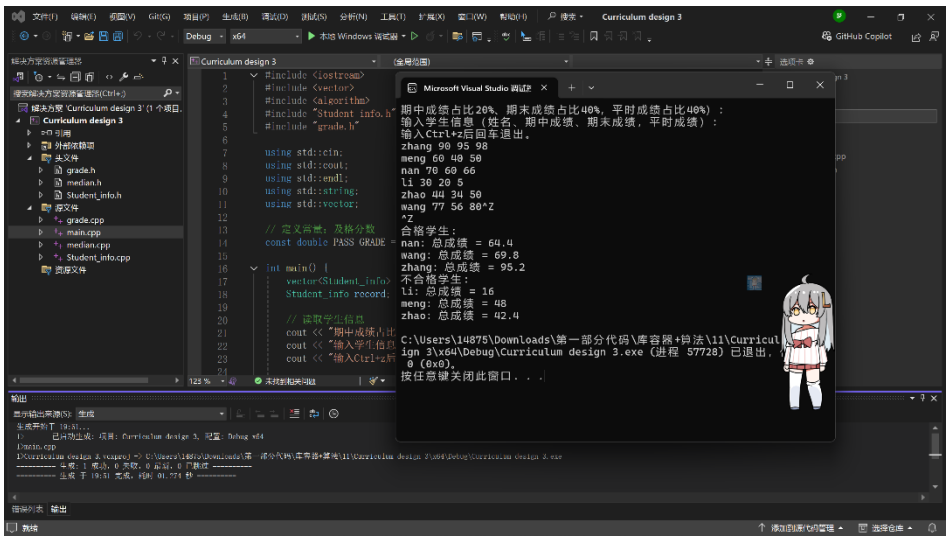

图 3.1 课程设计三程序运行示例

五、附完整代码：

main.cpp：

```
#include <iostream>
#include <vector>
#include <algorithm>
#include "Student_info.h"
#include "grade.h"

using std::cin;
using std::cout;
using std::endl;
using std::string;
using std::vector;

// 定义常量：及格分数
const double PASS_GRADE = 60.0;

int main() {
```

```
    vector<Student_info> students;
    Student_info record;

    // 读取学生信息
    cout << "期中成绩占比 20%、期末成绩占比 40%，平时成绩占比 40%）:" << endl;
    cout << "输入学生信息（姓名、期中成绩、期末成绩，平时成绩）:" << endl;
    cout << "输入 Ctrl+z 后回车退出。" << endl;

    while (read(cin, record)) {
        students.push_back(record);
    }

    // 将学生分为及格和不及格两组
    vector<Student_info> pass, fail;
    for (const auto& student : students) {
        if (grade(student) >= PASS_GRADE) {
            pass.push_back(student);
        }
        else {
            fail.push_back(student);
        }
    }

    // 按名字排序
    std::sort(pass.begin(), pass.end(), compare);
    std::sort(fail.begin(), fail.end(), compare);

    // 输出及格学生信息
    cout << "合格学生:" << endl;
    for (const auto& student : pass) {
        cout << student.name << ": 总成绩 = " << grade(student) << endl;
    }

    // 输出不及格学生信息
    cout << "不合格学生:" << endl;
    for (const auto& student : fail) {
        cout << student.name << ": 总成绩 = " << grade(student) << endl;
    }

    return 0;
```

```
}
```

student_info.h:

```
#ifndef GUARD_Student_info
#define GUARD_Student_info

// `Student_info.h' header file
#include <iostream>
#include <string>
#include <vector>

struct Student_info {
    std::string name;
    double midterm, final;
    std::vector<double> homework;
};

bool compare(const Student_info&, const Student_info&);
std::istream& read(std::istream&, Student_info&);
std::istream& read_hw(std::istream&, std::vector<double>&);
#endif
```

student_info.cpp:

```
// source file for `Student_info'-related functions
#include "Student_info.h"

using std::istream;  using std::vector;

bool compare(const Student_info& x, const Student_info& y)
{
    return x.name < y.name;
}

istream& read(istream& is, Student_info& s)
{
    // read and store the student's name and midterm and final exam g
rades
    is >> s.name >> s.midterm >> s.final;

```

```
    read_hw(is, s.homework);  // read and store all the student's hom
ework grades
    return is;
}

// read homework grades from an input stream into a `vector<double>'

istream& read_hw(istream& in, vector<double>& hw)
{
    if (in) {
        // get rid of previous contents
        hw.clear();

        // read homework grades
        double x;
        while (in >> x)
            hw.push_back(x);

        // clear the stream so that input will work for the next stud
ent
        in.clear();
    }
    return in;
}
```

median.h:

```
#ifndef GUARD_median_h
#define GUARD_median_h

// `median.h'--final version
#include <vector>
double median(std::vector<double>);

#endif
```

median.cpp:

```
// source file for the `median' function
#include <algorithm>    // to get the declaration of `sort'
#include <stdexcept>    // to get the declaration of `domain_error'
#include <vector>       // to get the declaration of `vector'
```

```
5.
6.  using std::domain_error;   using std::sort;   using std::vector;
7.
8.  #include "median.h"
9.
10. // compute the median of a `vector<double>'
11. // note that calling this function copies the entire argument `vector
    '
12. double median(vector<double> vec)
13. {
14. #ifdef _MSC_VER
15.     typedef std::vector<double>::size_type vec_sz;
16. #else
17.     typedef vector<double>::size_type vec_sz;
18. #endif
19.
20.     vec_sz size = vec.size();
21.     if (size == 0)
22.         throw domain_error("median of an empty vector");
23.
24.     sort(vec.begin(), vec.end());
25.
26.     vec_sz mid = size/2;
27.
28.     return size % 2 == 0 ? (vec[mid] + vec[mid-1]) / 2 : vec[mid];
29. }
```

grade. h:

```
1.  #ifndef GUARD_grade_h
2.  #define GUARD_grade_h
3.
4.  #include <vector>
5.  #include "Student_info.h"
6.
7.  double grade(double, double, double);
8.  double grade(double, double, const std::vector<double>&);
9.  double grade(const Student_info&);
10.
11. bool pgrade(const Student_info&);
12. bool fgrade(const Student_info&);
13.
14. #endif
```

grade. cpp:

```cpp
#include <stdexcept>
#include <vector>
#include "grade.h"
#include "median.h"
#include "Student_info.h"

using std::domain_error;  using std::vector;


// compute a student's overall grade from midterm and final exam grades and homework grade
double grade(double midterm, double final, double homework)
{
    return 0.2 * midterm + 0.4 * final + 0.4 * homework;
}

// compute a student's overall grade from midterm and final exam grades
// and vector of homework grades.
// this function does not copy its argument, because `median' does so for us.
double grade(double midterm, double final, const vector<double>& hw)

{
    if (hw.size() == 0)
        throw domain_error("student has done no homework");
    return grade(midterm, final, median(hw));
}

double grade(const Student_info& s)
{
    return grade(s.midterm, s.final, s.homework);
}

// predicate to determine whether a student failed
bool fgrade(const Student_info& s)
{
    return grade(s) < 60;
}

```

```
37. bool pgrade(const Student_info& s)
38. {
39.     return !fgrade(s);
40. }
```

评分表

序号	内容	得分
1	能利用标准库进行编程，库函数使用正确。（**0.5**）	
2	软件架构合理，功能模块划分清晰，代码编写规范，结果正确。（**0.5**）	
	总计	

评分人：

日期：