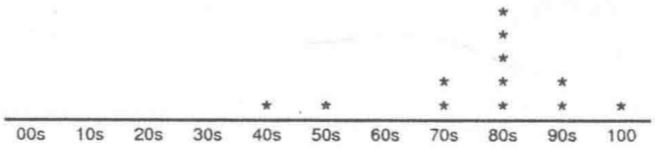
习题一~习题二

4. **直方图**是一个图，它通过把数据划分进单独的范围，然后指出每个范围内中有多少个数据值落入该范围的方式来显示一组值。例如，给出一组考试分数：

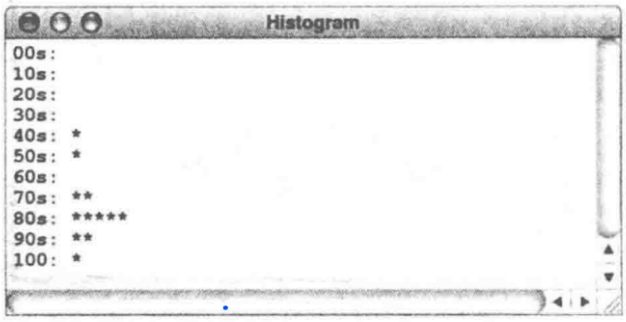
100，95，47，88，86，92，75，89，81，70，55，80

一个传统的直方图有如下形式：



直方图中的星号表示：有一个分数在 40 ～ 49 之间，有一个分数在 50 ～ 59 之间，有五个分数在 80 ～ 89 之间，依此类推。

然而，当你用计算机生成直方图的时候，将其放在一页的一边是很简单的，如以下这个示例运行结果：



编写一个程序，这个程序从一个数据文件中向一个元素类型为整型的 Vector 对象中输入数据，然后在一个直方图中显示这些数字，这个直方图被划分的范围依次是 0 ～ 9、10 ～ 19、20 ～ 29，依此类推，直到只包含值 100 的范围。你的程序应该尽可能产生和示例程序一样的输出。

```
void render(const vector<int> &rhs) {
    vector<string> score_rank {"00s: ", "10s: ", "20s: ", "30s: ",
                               "40s: ", "50s: ", "60s: ", "70s: ",
                               "80s: ", "90s: ", "100: " };
    for (auto i : rhs) {
        score_rank[i / 10] += "*";
    }

    for (auto i : score_rank) {
        cout << i << endl;
    }

}
// file: 100, 95, 47, 88, 86, 92, 75, 89, 81, 70, 55, 80
// 将文件中的数据转换成一个vector<int>的数组
void lineToNumberArray(vector<int> &result, string line) {
   int start = -1;
   for (int i = 0; i < line.length(); i++) {
      char ch = line[i];
      if (isdigit(ch)) {
```

```
            if (start == -1) start = i;
        } else {
            if (start >= 0) {
                result.push_back(atoi(line.substr(start, start - i).c_str()));
                start = -1;
            }
        }
    }
    if (start >= 0) result.push_back(atoi((line.substr(start).c_str())));
}
```

```
int main() {
    const string kfile_name = "exam_score.txt";
    ifstream infile{kfile_name};
    vector<int> res{};
    string line;

    while (getline(infile, line)) {
        lineToNumberArray(res, line);
    }

    render(res);
    cout << "---------------" << endl;
    render(0, 100, 10, res);
    return 0;
}
```

5. 通过定义一个名为 hist.h 的接口来扩展之前习题的灵活性，这个接口给出了用户对于直方图的形式的更多的控制。你的接口的最低要求是：应该允许用户指定最大值和最小值，以及每个直方图范围的大小，但你也可以额外的添加一些其他的功能。

```
// 实现思路:
//    while (l <= high) {
//        rank[l] += integerToString(l) + "s: ";
//        l += range;
//    }
// 以range为分割进行初始化，将rank初始化成 类似于    vector<string> score_rank {"00s: ", "10s: ", "20s: ", "30s: ","40s: ", "50s: ", "60s: ", "70s: ","80s: ", "90s: ", "100: " };的形式，  同时，为了避免使用线性结构导致数组过于稀疏，因此采用了map作为容器。
// 之后，通过floor(val, range)，将值进行向range取整，进而作为key访问map.
void render(int low, int high, int range, const vector<int> &rhs) {
    map<int, string> rank {};
    int l = low;

    while (l <= high) {
        rank[l] += integerToString(l) + "s: ";
        l += range;
    }

    for (auto i : rhs) {
        rank[floor(i, range)] += "*";
```

```
    }

    for (auto [key, val] : rank) {
        cout << val << endl;
    }
}
string integerToString(int value) {
    ostringstream oss;
    oss << value;
    // 暂不考虑失败?
    return oss.str();
}
int floor(int i, int range) {
    return int(i / range) * range;
}
```

习题三

13. 现在是第一个
将变成最后一个
就时间而言，它们是一个变革的时代。
——鲍勃·迪伦，“变革的时代（*The Times They Are a-Changin*）”，1963
从鲍勃·迪伦的歌曲中获得灵感，编写一个函数：

```
void reverseQueue(Queue<string> & queue);
```

颠倒一个队列中的元素。记住你没有接触过队列的内部表示方法，因此，你必须想出一个算法（假定涉及其他的结构）用于完成这个任务。

莫名其妙，queue问题很大。

```cpp
#include <iostream>
#include <queue>
#include <vector>

using namespace std;

// 颠倒队列中的元素
void reverseQueue(queue<string> &queue) {
    vector<string> tempVector;

    // 将队列中的元素逐个压入向量中
    while (!queue.empty()) {
        tempVector.push_back(queue.front());
        queue.pop();
    }

    // 将向量中的元素逐个弹出并压回队列中（颠倒顺序）
    for (int i = tempVector.size() - 1; i ⩾ 0; --i) {
        queue.push(tempVector[i]);
    }
}

int main() {
    queue<string> myQueue;

    // 向队列中添加一些元素
    myQueue.push("A");
    myQueue.push("B");
    myQueue.push("C");
    myQueue.push("D");

    // 打印原始队列中的元素（不改变队列内容）
    queue<string> originalQueue = myQueue; // 备份原始队列
    cout << "Original Queue:" << endl;
    while (!originalQueue.empty()) {
        cout << originalQueue.front() << " ";
        originalQueue.pop();
```

```
    }
    cout << endl;

    // 将队列中的元素颠倒
    reverseQueue(myQueue);

    // 打印颠倒后的队列中的元素
    cout << "Reversed Queue:" << endl;
    while (!myQueue.empty()) {
        cout << myQueue.front() << " ";
        myQueue.pop();
    }
    cout << endl;

    return 0;
}
```

习题四

编写一个程序显示一个表格。它根据单词的长度进行排序，并展示了不同长度的单词在英文字典中出现的次数。

```cpp
#include <iostream>
#include <map>

using namespace std;

int main() {
    // 定义一个map来存储单词长度及其出现的次数
    map<int, int> wordLengthFrequency;

    // 输入一些单词，以0结束输入
    string word;
    cout << "Enter words (enter 0 to stop):" << endl;
    while (true) {
        cin >> word;
        if (word == "0") {
            break;
        }
        wordLengthFrequency[word.length()]++;
    }

    // 输出表头
    cout << "Word Length\tFrequency" << endl;

    // 输出map中的内容，map已经按照键的大小排序
    for (const auto& pair : wordLengthFrequency) {
        cout << pair.first << "\t\t" << pair.second << endl;
    }

    return 0;
}
```