**第五次实验**

**注意事项**

(1) 不是每个作业题都需要提交，注意审题

**实验九、群体数据与流**

(1) 观察以下程序的输出，注意对输出格式的控制方法。（不需要提交）

#include < fstream>

using namespace std;

#define D(a) T << #a << endl; a

ofstream T("output.out");

int main()

{

D(int i = 53;)

D(float f = 4700113.141593;)

char \*s = "Is there any more?";

D(T.setf(ios::unitbuf);)

D(T.setf(ios::showbase);)

D(T.setf(ios::uppercase);)

D(T.setf(ios::showpos);)

D(T << i << endl;)

D(T.setf(ios::hex, ios::basefield);)

D(T << i << endl;)

D(T.unsetf(ios::uppercase);)

D(T.setf(ios::oct, ios::basefield);)

D(T << i << endl;)

D(T.unsetf(ios::showbase);)

D(T.setf(ios::dec, ios::basefield);)

D(T.setf(ios::left, ios::adjustfield);)

D(T.fill('0');)

D(T << "fill char: " << T.fill() << endl;)

D(T.width(8);)

T << i << endl;

D(T.setf(ios::right, ios::adjustfield);)

D(T.width(8);)

T << i << endl;

D(T.setf(ios::internal, ios::adjustfield);)

D(T.width(8);)

T << i << endl;

D(T << i << endl;) //Without width(10)

D(T.unsetf(ios::showpos);)

D(T.setf(ios::showpoint);)

D(T << "prec= " << T.precision() << endl;)

D(T.setf(ios::scientific, ios::floatfield);)

D(T << endl << f << endl;)

D(T.setf(ios::fixed, ios::floatfield);)

D(T << f << endl;)

D(T.setf(0, ios::floatfield);)

D(T << f << endl;)

D(T.precision(16);)

D(T << "prec= " << T.precision() << endl;)

D(T << endl << f << endl;)

D(T.setf(ios::scientific, ios::floatfield);)

D(T << endl << f << endl;)

D(T.setf(ios::fixed, ios::floatfield);)

D(T << f << endl;)

D(T.setf(0, ios::floatfield);)

D(T << f << endl;)

D(T.width(8);)

T << s << endl;

D(T.width(36);)

T << s << endl;

D(T.setf(ios::left, ios::adjustfield);)

D(T.width(36);)

T << s << endl;

D(T.unsetf(ios::showpoint);)

D(T.unsetf(ios::unitbuf);)

}

(2) 阅读并运行以下程序，领会容器的使用、函数指针的使用，感受专业排序算法的威力。（不需要提交）

#include <vector>

#include <algorithm>

#include <cstdlib>

#include <time.h>

#include <iostream>

using namespace std;

struct Student {

int id;

int age;

bool sex;

char name[64];

};

bool compareStudent(Student &s1, Student &s2) {

return s1.id < s2.id;

}

Student arr[40000];

int main() {

vector<Student> v;

clock\_t time0, time1, time2, time3;

time0 = clock();

for (int i = 0; i < 40000; i++) {

arr[i].id = rand();

arr[i].age = 20;

arr[i].sex = (arr[i].id & 1) ? true : false;

v.push\_back(arr[i]);

}

time1 = clock();

sort(v.begin(), v.end(), compareStudent);

time2 = clock();

for (int i = 0; i < 39999; i++) {

Student tmp;

int max = 0;

for (int j = 0; j < 39999 - i; j++) {

if (!compareStudent(arr[j], arr[max]))

max = j;

}

tmp = arr[39999 - i];

arr[39999 - i] = arr[max];

arr[max] = tmp;

}

time3 = clock();

cout << "Create and insert: " << time1 - time0 << "ms" << endl;

cout << "Sort by vector: " << time2 - time1 << "ms" << endl;

cout << "Sort by selection: " << time3 - time2 << "ms" << endl;

return 0;

}

思考：如果把compareStudent()改成如下，意味着什么？

bool compareStudent(Student &s1, Student &s2) {

return s1.sex == s2.sex ? s1.id < s2.id : s1.sex < s2.sex;

}

(3) 在第3次实验时，曾针对Student结构设计了Deque容器或Hashset容器。但是该容器仅适用于Student结构。请利用模板技术，对当时写的程序进行改造，使容器能灵活适配各种类型的元素。

说明：

1）相关代码已由老师提供，请填充其中空缺的部分，不要修改老师写好的程序。

2）Hashset内部是链表，要求元素具备next指针，但既然要灵活适配各种类型，就不能要求元素具备next指针，为此，设计了Node类对元素做一层包装并提供next指针，Hashset中真正的“零件”是Node类对象。改造程序时，时刻要分清楚，你在操作的究竟是“零件”还是“零件”内包装的“元素”。

(4) 在老师提供的程序中填空，打开老师提供的文本文件，利用multiset容器读入所有姓名，然后按字母顺序输出到另一个文本文件，输出时在每一行前加行号。

(5) 编写程序，实现文件拷贝功能。程序运行时，通过控制台接受用户输入两个参数，即源文件和目标文件，命令格式如下：

cp.exe source.file destination.file

提示：

1）以二进制形式操作文件，对于源文件使用ios\_base::\_Nocreate方式，表示不新建文件（文件不存在则出错），对于目标文件使用ios\_base::\_Noreplace方式，表示不覆盖文件（文件已存在则出错）。

2）循环读取文件时，最后一次读取读到文件尾，通常不会恰好填满缓冲区，可以通过ifstream::gcount()获取刚才读取的字节数。

3）int main(int argc, char \*\*argv)可以让程序获取用户输入的参数。

运行效果如下：

