Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Пермский национальный исследовательский политехнический университет» (ПНИПУ)

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

**Лабораторная работа «Классы №13»**

**по дисциплине**

**«Основы алгоритмизации и программирования»**

**(Семестр** 2)

Выполнил студент гр. ИВТ-21-1б

Гребнев Алексей Дмитриевич

Проверил:

Яруллин Денис Владимирович

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

(оценка) (подпись)

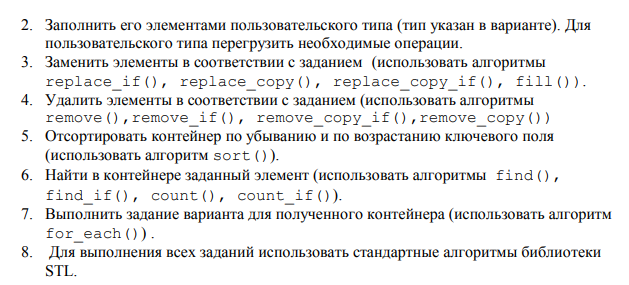
\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

г.Пермь -2022

**Постановка задачи 1**





**Исходные данные**

Вариант 10:



**Код-программы**

**1)Главная часть:**

##include <iostream>

#include "Money.h"

#include <vector>

#include <algorithm>

using namespace std;

typedef vector<Money> Tvector;

Money s;

struct Greater\_s

{

    bool operator()(Money m)

    {

        if(m>s) {return true;}

        else {return false;}

    }

};

Tvector make\_vector(int n)

{

    Money a;

    Tvector v;

    for(int i = 0; i<n; i++)

    {

        cin >> a;

        v.push\_back(a);

    }

}

void print\_vector(Tvector v)

{

    for (int i = 0; i<v.size();i++)

    {

        cout << v[i] << " ";

    }

    cout << endl;

}

Money srednee(Tvector v)

{

    Money m = v[0];

    //перебор вектора

    for(int i = 1; i < v.size(); i++)

    {

        m=m+v[i];

    }

    int n = v.size();//кол-во элементов в векторе

    return m/n;

}

struct Comp\_less //предикат для сортировки по убыванию (т.е его изменения)

{

    public:

        bool operator()(Money m1,Money m2)

        {

            if (m1 > m2) return true;

            else return false;

        }

};

struct Equal\_s//предикат для поиска заданного значения

{

    bool operator() (Money m)

    {

        return m == s;

    }

};

void del(Money& m)

{

    m = m / s;

}

int main()

{

    int n;

    cout << "Enter N: ";

    cin >> n;

    Tvector v;

    v = make\_vector(n);

    print\_vector(v);

    Tvector :: iterator i;

    //поставим итератор на макс элемент

    i = max\_element(v.begin(),v.end());

    cout << "max= " << \*(i) << endl;

    Money k = \*(i);

    s = srednee(v);//нашли сред арифм вектора

    cout << "srednee vect= "<<s<<endl;

    //замена с использованием предиката

    replace\_if(v.begin(),v.end(),Greater\_s(),k);

    cout << "Zamena: " << endl;

    print\_vector(v);

    cout << "Sort po ubiv: " << endl;

    sort(v.begin(),v.end(),Comp\_less());

    print\_vector(v);

    cout << "Sort po vozrst: " << endl;

    sort(v.begin(),v.end());

    print\_vector(v);

    cout << "Poisk: " << endl;

    cin >> s;

    //поиск элементов, удовлетворяющих условию предиката

    i = find\_if(v.begin(),v.end(),Equal\_s());

    if( i != v.end()) //если нет конца вектора

    {

        cout << \*(i) << endl;

    }

    else

    {

        cout << "ERROR POISK!" << endl;

    }

    cout << "Delete\_move: "<<endl;

    i = min\_element(v.begin(),v.end());

    s=\*i;

    //переместим элементы совпадающие с min в конец вектора

    i=remove\_if(v.begin(),v.end(),Equal\_s());

    //удалить все элементы, начиная с i и до конца вектора

    v.erase(i,v.end());

    print\_vector(v);

    cout << "Delenie: " << endl;

    i=max\_element(v.begin(),v.end());

    s=\*i;

    //для каждого элемента вектора вызывается функция del

    for\_each(v.begin(),v.end(),del);//выполнение какого-то действия с каждым элементом диапазона по отдельности

    print\_vector(v);

}

**2)Заголовочный файл:**

#pragma once

#include <iostream>

using namespace std;

class Money

{

private:

    long rub;

    int cop;

public:

    Money();

    Money(long k, int p);

    Money(const Money& g);

    ~Money();

    long get\_rub();

    int get\_cop();

    void set\_rub(long k);

    void set\_cop(int p);

    void show\_money();

    Money& operator-(const Money&);

    Money& operator+(const Money&);

    Money operator=(const Money&);

    Money& operator/(Money&);

    Money& operator/(int);

    Money& operator\*(const Money&);

    bool operator!=(const Money&);

    bool operator==(const Money&);

    bool operator<(const Money&);

    bool operator>(const Money&);

    friend ostream& operator<<(ostream& out, const Money& m);

    friend istream& operator>>(istream& in, Money& m);

};

#include <iostream>

#include "Money.h"

using namespace std;

    Money :: Money()

    {

        rub = 0;

        cop = 0;

    }

    Money :: Money(long k, int p)

    {

        rub = k;

        cop = p;

    }

    Money :: Money(const Money& g)

    {

        rub = g.rub;

        cop = g.cop;

    }

    Money :: ~Money()

    {

    }

    long Money :: get\_rub()

    {

        return rub;

    };

    int Money :: get\_cop()

    {

        return cop;

    };

    void Money :: set\_rub(long k)

    {

        rub = k;

    };

    void Money :: set\_cop(int p)

    {

        cop = p;

    };

    void Money :: show\_money()

    {

        cout <<  "Рубли: " << rub << endl << "Копейка: " << cop << endl;

    }

    Money& Money :: operator+(const Money& m)

    {

        long rez;

        rez = (rub \* 100 + cop) - (m.rub \* 100 + m.cop);

        rub = rez / 100; //целое

        cop = rez % 100; //остаток

        return \*this;

    }

    Money& Money :: operator/(Money& m)

    {

        long rez;

        rez = (rub \* 100 + cop) / (m.rub\*100 + m.cop);

        rub = rez / 100;//целое

        cop = rez % 100;//остаток

        return \*this;

    }

    Money& Money :: operator/(int m)

    {

        long rez;

        rez = (rub \* 100 + cop) / m;

        rub = rez / 100;//целое

        cop = rez % 100;//остаток

        return \*this;

    }

    bool Money :: operator==(const Money& m1)

    {

    if ((rub == m1.rub) && (cop == m1.cop)) {return true;}

    else{return false;}

    }

    bool Money :: operator!=(const Money& m1)

    {

    if ((rub != m1.rub) && (cop != m1.cop)) {return true;}

    else{return false;}

    }

    bool Money :: operator<(const Money& m1)

    {

        if(rub < m1.rub) {return true;}

        if(rub == m1.rub && cop < m1.cop)

        {return true;}

        return false;//если ничего не сработало!

    }

    bool Money :: operator>(const Money& m1)

    {

        if(rub > m1.rub) {return true;}

        if(rub == m1.rub && cop > m1.cop)

        {return true;}

        return false;//если ничего не сработало

    }

    istream& operator>>(istream& in, Money& tmp)

    {

    cout << "Введите кол-во рублей: "; in >> tmp.rub;

    cout << "Введите кол-во копеек: "; in >> tmp.cop;

    return in;

    }

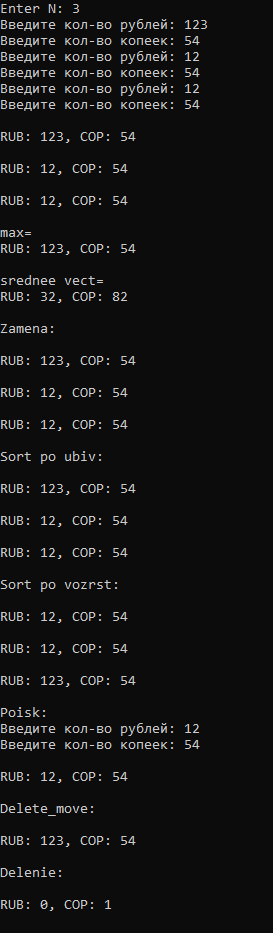
    ostream& operator<<(ostream& out, const Money& tmp)

    {

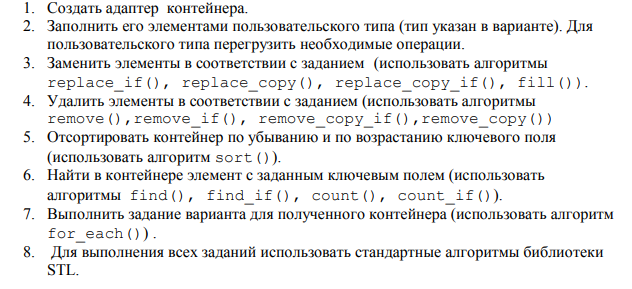
    return (out << endl << "RUB: " << tmp.rub << ", COP: " << tmp.cop << endl);

    }

**Результат**



**Постановка задачи 2**

****

**Исходные данные**

Вариант 10

****

**Код-программы  
  
1)Главная часть:**

#include <iostream>

#include <algorithm>

#include <queue>

#include <vector>

#include "Money.h"

using namespace std;

typedef priority\_queue <Money> Queue;

typedef vector <Money> Vector;

Money s;

struct Greater\_s

{

    bool operator()(Money m)

    {

        if(m>s) {return true;}

        else {return false;}

    }

};

Queue make\_queue(int n)

{

    Money a;

    Queue q;

    for(int i = 0; i<n; i++)

    {

        cin >> a;

        q.push(a);

    }

}

Vector q\_to\_v(Queue st)

{

    Vector v;

    while (!st.empty())

    {

        v.push\_back(st.top());

        st.pop();

    }

    return v;

}

Queue v\_to\_q(Vector v)

{

    Queue q;

    for(int i = 0; i < v.size(); i++)

    {

        q.push(v[i]);

    }

    return q;

}

void print\_queue(Queue st)

{

    Vector v\_tmp;

    v\_tmp = q\_to\_v(st);

    while(!st.empty())

    {

        cout << st.top() << endl;

        st.pop();

    }

    st = v\_to\_q(v\_tmp);

}

Money srednee(Vector v)

{

    Money m = v[0];//сумма

    //перебор вектора

    for(int i = 1; i < v.size(); i++)

    {

        m=m+v[i];

    }

    int n = v.size();//кол-во элементов в векторе

    return m/n;

}

struct Comp\_less //предикат для сортировки по убыванию (т.е его изменения)

{

    public:

        bool operator()(Money m1,Money m2)

        {

            if (m1 > m2) return true;

            else return false;

        }

};

struct Equal\_s//предикат для поиска заданного значения

{

    bool operator() (Money m)

    {

        return m == s;

    }

};

void del(Money& m)

{

    m = m / s;

}

int main()

{

    setlocale(LC\_ALL,"RUS");

    int n;

    cout << "Введите значение n: ";

    cin >> n;

    Queue q = make\_queue(n);

    print\_queue(q);

    Vector v = q\_to\_v(q);

    Vector :: iterator i;

    //поставим итератор на макс элемент

    i = max\_element(v.begin(),v.end());

    cout << "max= " << \*(i) << endl;

    Money k = \*(i);

    s = srednee(v);//нашли сред арифм вектора

    cout << "srednee vect= "<<s<<endl;

    //замена с использованием предиката

    replace\_if(v.begin(),v.end(),Greater\_s(),k);

    cout << "Zamena: " << endl;

    q = v\_to\_q(v);

    print\_queue(q);

    v = q\_to\_v(q);

    cout << "Sort po ubiv: " << endl;

    sort(v.begin(),v.end(),Comp\_less());

    q = v\_to\_q(v);

    print\_queue(q);

    v = q\_to\_v(q);

    cout << "Sort po vozrst: " << endl;

    sort(v.begin(),v.end());

    q = v\_to\_q(v);

    print\_queue(q);

    v = q\_to\_v(q);

    cout << "Poisk: " << endl;

    cin >> s;

    //поиск элементов, удовлетворяющих условию предиката

    i = find\_if(v.begin(),v.end(),Equal\_s());

    if( i != v.end()) //если нет конца вектора

    {

        cout << \*(i) << endl;

    }

    else

    {

        cout << "ERROR POISK!" << endl;

    }

    cout << "Delete\_move: "<<endl;

    i = min\_element(v.begin(),v.end());

    s=\*i;

    //переместим элементы совпадающие с min в конец вектора

    i=remove\_if(v.begin(),v.end(),Equal\_s());

    //удалить все элементы, начиная с i и до конца вектора

    v.erase(i,v.end());

    q = v\_to\_q(v);

    print\_queue(q);

    v = q\_to\_v(q);

    cout << "Delenie: " << endl;

    i=max\_element(v.begin(),v.end());

    s=\*i;

    //для каждого элемента вектора вызывается функция del

    for\_each(v.begin(),v.end(),del);//выполнение какого-то действия с каждым элементом диапазона по отдельности

    q = v\_to\_q(v);

    print\_queue(q);

}

**2)Заголовочный файл**

#pragma once

#include <iostream>

using namespace std;

class Money

{

private:

    long rub;

    int cop;

public:

    Money();

    Money(long k, int p);

    Money(const Money& g);

    ~Money();

    long get\_rub();

    int get\_cop();

    void set\_rub(long k);

    void set\_cop(int p);

    void show\_money();

    Money& operator-(const Money&);

    Money& operator+(const Money&);

    Money operator=(const Money&);

    Money& operator/(Money&);

    Money& operator/(int);

    Money& operator\*(const Money&);

    bool operator!=(const Money&);

    bool operator==(const Money&);

    bool operator<(const Money&);

    bool operator>(const Money&);

    friend ostream& operator<<(ostream& out, const Money& m);

    friend istream& operator>>(istream& in, Money& m);

};

#include <iostream>

#include "Money.h"

using namespace std;

    Money :: Money()

    {

        rub = 0;

        cop = 0;

    }

    Money :: Money(long k, int p)

    {

        rub = k;

        cop = p;

    }

    Money :: Money(const Money& g)

    {

        rub = g.rub;

        cop = g.cop;

    }

    Money :: ~Money()

    {

    }

    long Money :: get\_rub()

    {

        return rub;

    };

    int Money :: get\_cop()

    {

        return cop;

    };

    void Money :: set\_rub(long k)

    {

        rub = k;

    };

    void Money :: set\_cop(int p)

    {

        cop = p;

    };

    void Money :: show\_money()

    {

        cout <<  "Рубли: " << rub << endl << "Копейка: " << cop << endl;

    }

    Money& Money :: operator+(const Money& m)

    {

        long rez;

        rez = (rub \* 100 + cop) - (m.rub \* 100 + m.cop);

        rub = rez / 100; //целое

        cop = rez % 100; //остаток

        return \*this;

    }

    Money& Money :: operator/(Money& m)

    {

        long rez;

        rez = (rub \* 100 + cop) / (m.rub\*100 + m.cop);

        rub = rez / 100;//целое

        cop = rez % 100;//остаток

        return \*this;

    }

    Money& Money :: operator/(int m)

    {

        long rez;

        rez = (rub \* 100 + cop) / m;

        rub = rez / 100;//целое

        cop = rez % 100;//остаток

        return \*this;

    }

    bool Money :: operator==(const Money& m1)

    {

    if ((rub == m1.rub) && (cop == m1.cop)) {return true;}

    else{return false;}

    }

    bool Money :: operator!=(const Money& m1)

    {

    if ((rub != m1.rub) && (cop != m1.cop)) {return true;}

    else{return false;}

    }

    bool Money :: operator<(const Money& m1)

    {

        if(rub < m1.rub) {return true;}

        if(rub == m1.rub && cop < m1.cop)

        {return true;}

        return false;//если ничего не сработало!

    }

    bool Money :: operator>(const Money& m1)

    {

        if(rub > m1.rub) {return true;}

        if(rub == m1.rub && cop > m1.cop)

        {return true;}

        return false;//если ничего не сработало

    }

    istream& operator>>(istream& in, Money& tmp)

    {

    cout << "Введите кол-во рублей: "; in >> tmp.rub;

    cout << "Введите кол-во копеек: "; in >> tmp.cop;

    return in;

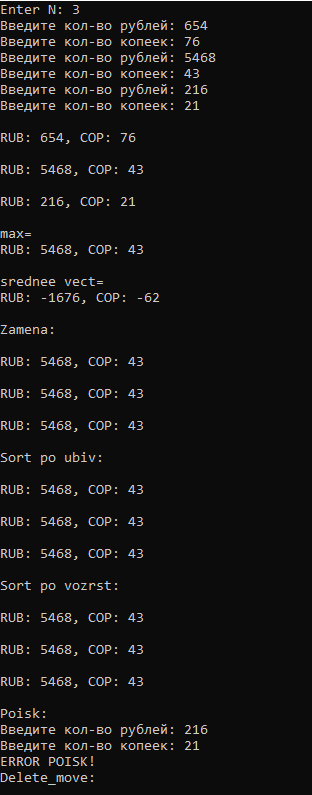
    }

    ostream& operator<<(ostream& out, const Money& tmp)

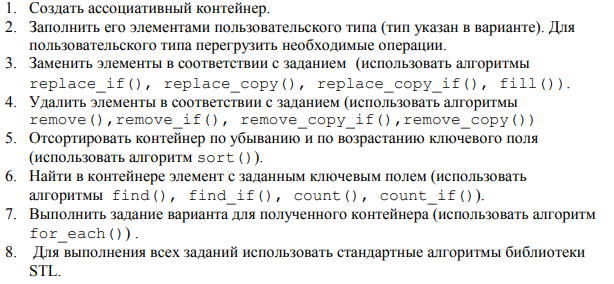
    {

    return (out << endl << "RUB: " << tmp.rub << ", COP: " << tmp.cop << endl);

    }

**Результат  
**

**Постановка задачи 3**

****

**Исходные данные**

Вариант 10



**1) Главная часть:**

#include <iostream>

#include "Money.h"

#include <vector>

#include <set>

using namespace std;

typedef multiset <Money> mySet;

typedef multiset<Money> :: iterator Iter; //для работы в действиях

Money s;

struct Greater\_s

{

    bool operator()(Money m)

    {

        if(m>s) {return true;}

        else {return false;}

    }

};

mySet make\_MulSet(int n)

{

    Money a;

    mySet mst;

    for(int i = 0; i<n; i++)

    {

        cin >> a;

        mst.insert(a);

    }

    return mst;

}

void print\_MulSet(mySet mst)

{

    Iter iter = mst.begin();

    while (iter != mst.end())

    {

        cout << \*iter;

        iter++;

    }

    cout << endl;

}

Money srednee(mySet mst)

{

    Iter iter = mst.begin();

    Money m;//сумма

    //перебор вектора

    int n = 0;

    while (iter != mst.end())

    {

        m=m+(\*iter);

        n++; //кол-во элементов в векторе

        iter++;

    }

    return m/n;

}

struct Comp\_less //предикат для сортировки по убыванию (т.е его изменения)

{

    public:

        bool operator()(Money m1,Money m2)

        {

            if (m1 > m2) return true;

            else return false;

        }

};

struct Equal\_s//предикат для поиска заданного значения

{

    bool operator() (Money m)

    {

        return m == s;

    }

};

void del(Money& m)

{

    m = m / s;

}

int main()

{

    setlocale(LC\_ALL,"RUS");

    mySet mt;

    mySet mt1;

    mt1 = make\_MulSet(6);

    mt1.swap(mt);

    int n = 6;

    mt = make\_MulSet(n);

    print\_MulSet(mt);

    //iter = max\_element(mt.begin(),mt.end());

    s = srednee(mt);

    cout << "srednee (multiset) = "<<s<<endl;

    print\_MulSet(mt);

    Iter iter = mt.begin();

    Money k = \*(iter);

    remove\_if(mt.begin(),mt.end(),Greater\_s());

    print\_MulSet(mt);

    cout << "Poisk: " << endl;

    cin >> s;

    //поиск элементов, удовлетворяющих условию предиката

    iter = find(mt.begin(),mt.end(),Equal\_s());

    if( iter != mt.end()) //если нет конца вектора

    {

        cout << \*(iter) << endl;

    }

    else

    {

        cout << "ERROR POISK!" << endl;

    }

    cout << "Delete\_move: "<<endl;

    iter = min(mt.begin(),mt.end());

    s=\*iter;

    //переместим элементы совпадающие с min в конец вектора

    iter=remove\_if(mt.begin(),mt.end(),Equal\_s());

    //удалить все элементы, начиная с i и до конца вектора

    mt.erase(iter,mt.end());

    print\_MulSet(mt);

}

**2)Заголовочный файл:**

#pragma once

#include <iostream>

using namespace std;

class Money

{

private:

    long rub;

    int cop;

public:

    Money();

    Money(long k, int p);

    Money(const Money& g);

    ~Money();

    long get\_rub();

    int get\_cop();

    void set\_rub(long k);

    void set\_cop(int p);

    void show\_money();

    Money& operator-(const Money&);

    Money& operator+(const Money&);

    Money operator=(const Money&);

    Money& operator/(Money&);

    Money& operator/(int);

    Money& operator\*(const Money&);

    bool operator!=(const Money&);

    bool operator==(const Money&);

    bool operator<(const Money&);

    bool operator>(const Money&);

    friend ostream& operator<<(ostream& out, const Money& m);

    friend istream& operator>>(istream& in, Money& m);

};

#include <iostream>

#include "Money.h"

using namespace std;

    Money :: Money()

    {

        rub = 0;

        cop = 0;

    }

    Money :: Money(long k, int p)

    {

        rub = k;

        cop = p;

    }

    Money :: Money(const Money& g)

    {

        rub = g.rub;

        cop = g.cop;

    }

    Money :: ~Money()

    {

    }

    long Money :: get\_rub()

    {

        return rub;

    };

    int Money :: get\_cop()

    {

        return cop;

    };

    void Money :: set\_rub(long k)

    {

        rub = k;

    };

    void Money :: set\_cop(int p)

    {

        cop = p;

    };

    void Money :: show\_money()

    {

        cout <<  "Рубли: " << rub << endl << "Копейка: " << cop << endl;

    }

    Money& Money :: operator+(const Money& m)

    {

        long rez;

        rez = (rub \* 100 + cop) - (m.rub \* 100 + m.cop);

        rub = rez / 100; //целое

        cop = rez % 100; //остаток

        return \*this;

    }

    Money& Money :: operator/(Money& m)

    {

        long rez;

        rez = (rub \* 100 + cop) / (m.rub\*100 + m.cop);

        rub = rez / 100;//целое

        cop = rez % 100;//остаток

        return \*this;

    }

    Money& Money :: operator/(int m)

    {

        long rez;

        rez = (rub \* 100 + cop) / m;

        rub = rez / 100;//целое

        cop = rez % 100;//остаток

        return \*this;

    }

    bool Money :: operator==(const Money& m1)

    {

    if ((rub == m1.rub) && (cop == m1.cop)) {return true;}

    else{return false;}

    }

    bool Money :: operator!=(const Money& m1)

    {

    if ((rub != m1.rub) && (cop != m1.cop)) {return true;}

    else{return false;}

    }

    bool Money :: operator<(const Money& m1)

    {

        if(rub < m1.rub) {return true;}

        if(rub == m1.rub && cop < m1.cop)

        {return true;}

        return false;//если ничего не сработало!

    }

    bool Money :: operator>(const Money& m1)

    {

        if(rub > m1.rub) {return true;}

        if(rub == m1.rub && cop > m1.cop)

        {return true;}

        return false;//если ничего не сработало

    }

    istream& operator>>(istream& in, Money& tmp)

    {

    cout << "Введите кол-во рублей: "; in >> tmp.rub;

    cout << "Введите кол-во копеек: "; in >> tmp.cop;

    return in;

    }

    ostream& operator<<(ostream& out, const Money& tmp)

    {

    return (out << endl << "RUB: " << tmp.rub << ", COP: " << tmp.cop << endl);

    }