

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по лабораторной работе №1
по дисциплине «АиСД»
Тема: Рекурсия

Студент гр. 8304

Порывай П.А

Преподаватель

Фиалковский М

Санкт-Петербург

2015

Цель работы.

Умение пользоваться рекурсивными функциями.

Задание

Вариант 19

$\Phi(a) = a$, если $\|a\| \leq 2$, $\Phi(a) = \Phi(b)\Phi(y)$ если $a = by$ $\|b\| = \|y\|$ $\|a\| > 2$,

$\Phi(a) = \Phi(bh)\Phi(hy)$ если $a = bhy$ $\|b\| = \|y\|$, $\|a\| > 2$, $\|h\| = 1$

Выполнение работы

Класс Line создан для записи векторов типа int, с перегрузкой операторов »
, «

Функция RecF принимает на вход вектор целых чисел и применяет к нему алгоритм описанный в задании. При каждом вызове выводится вектор, который был в аргументе при этом выполняется отступ по глубине. Если длина вектора делится на 2 и не равна 2 нужно вызвать 2 функции RecF. Как первая так и вторая функции в конструкции вида `(length % 2 == 0 && length != 2) {...}` дают определенную глубину(в них еще могут быть вызваны функции). Аналогично для конструкции где длина вектора делится на 2. При длине 1 или 2 часть возвращающегося вектора записывается в строку.

В main выводится диалоговое окно предлагающее считать данные из файла или из консоли

Выводы.

Получены навыки работы с рекурсивными функциями, вводом последовательности неизвестной длины из файла.

Приложение А. Исходный код

```
#include<iostream>
#include<vector>
#include<iterator> // ostream_iterator..
#include<string> // getline()
#include <sstream> // istreamstringstream()
#include<fstream> // ifstream()

using namespace std;

class Line {
    vector<int> _data; // vector стандартный шаблон C++ инициализируем пустой
    вектор вместо int можно поставить свой класс
public:
    const vector<int>& data() const // возвращаем вектор типа int. Возвращаем
    адрес вектора?
    {
        return _data;
    }
    friend istream& operator>>(istream& is, Line& line); // istream класс
    friend ostream& operator<<(ostream& os, const Line& line);
};

ostream& operator<<(ostream& os, const Line& line) // тип возвращаемого значения -
    объект класса ostream
{
    copy(line._data.begin(), line._data.end(), ostream_iterator<int>{os, "
    "}); // копирует из одного контейнера в другой здесь другой это os, класс шаблона
    ostream_iterator описывает объект итератора вывода
```

```

        return os;
    }

istream& operator>>(istream& is, Line& line)
{
    string str;
    getline(is, str);
    istringstream ss{ str };//Превращаем строку в поток
    line._data.assign(istream_iterator<int>{ss},
    {});//istream_iterator<int>{data} читать последовательность типа int (делитель "
    ")

    return is;
}

void RecF(vector<int> &vectorin, string* vectorout, int ident, int length) {

    int i ,j;

    for ( i = 0; i < ident; i++)
        cout << "\t";

    cout << "RecF(";

    for (i = 0; i < length; i++)
        cout << vectorin[i]<<" ";

    cout << ")" <<endl;

    if (length % 2 == 0 && length != 2) {

        vector<int> vectorcopy1(length / 2);
        vector<int> vectorcopy2(length / 2);

        for (i = 0; i < length / 2; i++)
            vectorcopy1[i] = vectorin[i];

        RecF(vectorcopy1, vectorout, ident + 1, length / 2);
    }
}

```

```

        j = 0;

        for (i = length / 2; i < length; i++) {
            vectorcopy2[j] = vectorin[i];
            j++;
        }

        RecF(vectorcopy2, vectorout, ident + 1, length / 2);

//    cout << "len" << length<<endl;
}
else if (length % 2 != 0 && length != 1) {
    //cout << "len" << length<<endl;
    vector<int> vectorcopy1((length / 2) + 1);
    vector<int> vectorcopy2((length / 2) + 1);

    for (i = 0; i < (length / 2) + 1; i++)
        vectorcopy1[i] = vectorin[i];

    RecF(vectorcopy1, vectorout, ident + 1, (length / 2) + 1);

    j = 0; //индекс для корректного копирования в новый массив

    for (i = (length / 2) ; i < length; i++) {
        vectorcopy2[j] = vectorin[i];
        j++;
    }
    //for (i = 0; i < (length / 2) + 1; i++)
        //cout << vectorcopy2[i];
    RecF(vectorcopy2, vectorout, ident + 1, (length / 2) + 1);

}
else if (length == 2) {

    //cout << "len2" << endl;
    *vectorout += to_string(vectorin[0]);
    *vectorout += " ";
    *vectorout += to_string(vectorin[1]);
    *vectorout += " ";

}
else if (length == 1) {

```

```

        *vectorout+= to_string(vectorin[0]);
    }

}

int main(){

    setlocale(LC_ALL, "Russian");
    string vector_out;
    vector_out = "";
    int ident = 0;
    int flag = 0;

    cout << "Ввод из файла или из консоли? (f , c)?\n";
    char arg;
    cin >> arg;
    if (arg == 'f') {

        ifstream
data("C:\\Users\\Павел\\source\\repos\\ConsoleApplication1\\Debug\\input\\inp.tx
t");

        vector<Line>                vectors_inp(istream_iterator<Line>{data},
{});//istream_iterator<Line>{data} - читать последовательность типа Line (те
каждая строка имеет тип Line, а разделитель \n, объект класса Line - data

        if (!data) {

            int len = vectors_inp.size();

            for (int i = 0; i < len; i++) {
                vector<int> vector_inp = vectors_inp[i].data();

                RecF(vector_inp, &vector_out, ident, vector_inp.size());
                cout << vector_out << endl;
                vector_out = "";
            }
        }
    }
}

```

```

        //cout << vectors_inp[i];
    }

}
else
    cout << "Файл не может быть открыт";

}
else if (arg == 'c') {

    string str;
    getline(cin, str); //иначе читает пустую строку

    while(getline(cin, str)){

        istringstream ss{ str }; //Превращаем строку в поток
        vector<int> vector_inp;
        vector_inp.assign(istream_iterator<int>{ss}, {});
        //cout << vector_inp.size();
        RecF(vector_inp, &vector_out, ident, vector_inp.size());
        cout << vector_out << endl;
        //Line vector_inpc;
        vector_out = "";
    }

}
else
    cout << "Такой команды нет";

return 0;

}

```

Тесты

Ввод

1 -23 4 5

12 3 4 -5 678

1 23 4

Вывод

1 -23 4 5

12 3 3 4 4 -5 -5 678

1 23 23 4