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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Алгоритмы и структуры данных»

Тема: Программирование рекурсивных алгоритмов.

Студент гр. 7383 _____ Тян Е.

Преподаватель _____ Размочаева Н. В.

Санкт-Петербург
2018

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Цель работы: ознакомиться с основными понятиями и приемами рекурсивного программирования, получить навыки программирования рекурсивных процедур и функций на языке программирования C++.

Формулировка задачи: реализовать рекурсивно функцию Ф преобразования целочисленного вектора α такую, что:

$$\Phi(\alpha)=\begin{cases}\alpha, \text{ если } |\alpha|\le 2,\\ \Phi(\beta)\Phi(\gamma), \text{ если } \alpha=\beta\gamma,\ |\beta|=|\gamma|,\ |\alpha|>2,\\ \Phi(\beta\, a)\,\Phi\,(a\,\gamma), \text{ если } \alpha=\beta a\gamma,\ |\beta|=|\gamma|,\ |\alpha|>2,\ |a|=1.\end{cases}$$

Входные данные: последовательность координат вектора.

2. РЕАЛИЗАЦИЯ ЗАДАЧИ

В данной работе используется главная функция (main()) и дополнительные функции (void divide_vector(int* vector, int size, int *myVector, int* pointer, int k), int *vector_modify(int* vector, int start, int end)). Параметры используемые в функции void divide_vector(int* vector, int size, int *myVector, int* pointer, int k):

- vector – вектор, который нужно обработать;
- size – длина вектора;
- myVector – новый вектор, куда будет записываться результат;
- pointer – указатель на последний элемент myVector;
- k – счетчик для более удобного представления вектора, при выводе на консоль.

Параметры используемые в функции int *vector_modify(int* vector, int start, int end):

- vector – исходный вектор;
- start – указатель начало вектора;
- end – указатель на конец нового вектора.

В функции main выводится меню на консоль, где можно выбрать число, соответствующее выполняемой операции. Считывается целое число и, при помощи switch(), выбирается необходимая опция. При выборе «1», пользователь вводит самостоятельно координаты вектора. При выборе «2», программа генерирует длину и координаты вектора в промежутке, введенном пользователем. При выборе «3» программа завершает работу. При выборе другого значения, программа выводит сообщение: «Проверьте введенные данные и повторите попытку» и ожидает дальнейших действий. Программа завершает свою работу только при выборе «3», в противном случае - программа будет ожидать дальнейших указаний. Если была выбрана опция генерирования значений, то после генерирования значений, на консоль выводятся полученные длина вектора и его координаты. Затем вызывается функция void divide_vector(int* vector, int size, int *myVector, int* pointer, int

k). Если пользователь ввел координаты вектора сам, то вызывается функция void divide_vector(int* vector, int size, int *myVector, int* pointer, int k). Далее на консоль выводится измененный вектор, и функция выполняет действия, в зависимости от выбора пользователя.

Рекурсивная функция void divide_vector(int* vector, int size, int *myVector, int* pointer, int k) делит вектор пополам. Сначала, на консоль выводится обрабатываемый вектор с табуляциями, в зависимости от номера вхождения вектора, что делается для удобного отслеживания действий программы. Затем, проверяется длина вектора: если она меньше, либо равна двум, то вектор остается без изменений. Если длина вектора больше двух, то длина проверяется на четность: если длина четная, то вектор делится пополам, и функция void divide_vector(int* vector, int size, int *myVector, int* pointer, int k) вызывает саму себя, чтобы делить вектор дальше, в противном случае, функция void divide_vector(int* vector, int size, int *myVector, int* pointer, int k) так же вызывает себя, но с измененными параметрами: координата, находящаяся посередине, дублируется. Также в функции вызывается функция int *vector_modify(int* vector, int start, int end) для изменения вектора. Как только был получен вектор длиной меньшей, либо равной двум, используется двумерный массив для записи частей вектора.

Функция int *vector_modify(int* vector, int start, int end) изменяет вектор, переписывая его.

Разберем для примера работы программы: вектор четной длины и вектор нечетной длины.

Если вектор четной длины, представленный на рис.1:

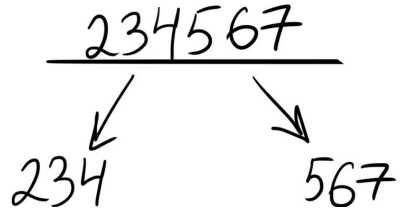

Рисунок 1 – Деление исходного
вектора четной длины на две части

Его длина 6. В функцию передастся первая половина и вторая половина поочереди.

1) divide_vector(vector_modify(vector,0,3),
3,myVector,pointer,2) – первая половина

2) divide_vector(vector_modify(vector,3,6),
3,myVector,pointer,2) – вторая половина

Далее, на вход программе поступили два вектора нечетной длины. Следовательно, нужно элемент, находящийся посередине, дублировать, что

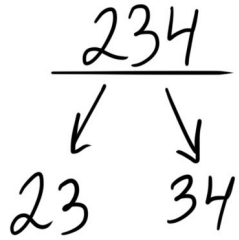

Рисунок 2 – Деление вектора
нечетной длины на две части

проиллюстрировано на рис. 2.

Тогда:

1) divide_vector(vector_modify(vector,0,2),2,myVector,
pointer,3) – первая половина с продублированной координатой

2) divide_vector(vector_modify(vector,1,size),2,myVector,
pointer,3) – вторая половина с продублированной координатой

На вход программе поступил вектор длины два, тогда в целочисленный массив myVector запишутся значения, поступившего на вход вектора.

Аналогичные действия происходят со второй половиной исходного вектора, как показано на рис. 3:

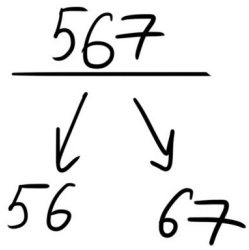

Рисунок 3 – Деление второй половины вектора нечетной длины на две части

В итоге на консоль будет выведет измененный вектор. Все действия, произошедший с вектором указаны на рис. 4:

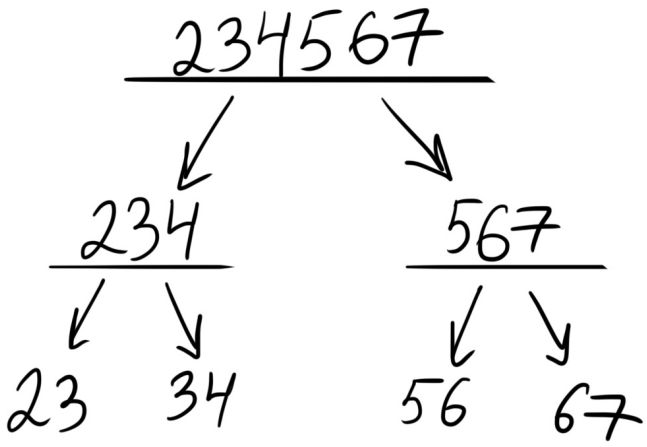

Рисунок 4 – Деление исходного вектора

3. ТЕСТИРОВАНИЕ

Программа была собрана в компиляторе GCC в OS Linux Ubuntu 12.04. Программа может быть скомпилирована с помощью команды:

g++ -Wall <имя_файла>.cpp

Тестовые случаи представлены в Приложении А.

Исходя из тестовых случаев можно увидеть, что в пятом тесте программа ведет себя неверно, поэтому была исправлена программа: добавлена проверка на корректность введенных данных – если среди введенных данных не было чисел, то программа выведет сообщение об ошибке: «Проверьте введенные данные и повторите попытку», но не пустую строку. Вывод исправленной программы указан в тестовых случаях №6, 7, 8.

После, тестовые случаи не выявили неверного поведения программы, что говорит о том, что по результатам тестирования было показано: поставленная задача была выполнена.

4. ВЫВОД

В ходе работы ознакомились с понятиями рекурсивного программирования. Была написана программа реализующая рекурсивную функцию Ф на языке программирования C++, преобразующую целочисленный вектор α так что:

$$\Phi(\alpha)=\begin{cases}\alpha, \text{если } |\alpha|\le 2,\\ \Phi(\beta)\Phi(\gamma), \text{если } \alpha=\beta\gamma,\ |\beta|=|\gamma|,\ |\alpha|>2,\\ \Phi(\beta\, a)\,\Phi\,(a\,\gamma),\ \text{если } \alpha=\beta a\gamma,\ |\beta|=|\gamma|,\ |\alpha|>2,\ |a|=1.\end{cases}$$

Было использовано генерирование случайных чисел. Разработано меню циклического взаимодействия с пользователем.

ПРИЛОЖЕНИЕ А. ТЕСТОВЫЕ СЛУЧАИ

№	Ввод	Вывод
1	5 7 8 9	5 7 8 9
2	6	6
3	5 7	5 7
4	4 5 7	4 5 5 7
5	Hvjhf bfs fj ygr	
6	Hvjhf bfs fj 3 54ygr	3 5 5 4
7		Проверьте введенные данные и повторите попытку
8	Hvjhf bfs fj ygr	Проверьте введенные данные и повторите попытку
9	6 (сгенерировано) 57 25 89 28 5 21	57 25 25 89 28 5 5 21

10	77 (сгенерировано) 35 78 84 86 81 25 45 54 38 73 18 88 72 64 30 35 17 98 38 7 74 46 57 5 43 79 28 72 89 53 39 78 56 18 10 20 20 19 58 74 12 65 46 66 30 93 41 52 86 18 93 53 56 56 24 40 36 49 50 35 53 28 9 99 74 19 64 97 80 14 98 26 17 16 59 69 46	35 78 78 84 84 86 86 81 25 45 45 54 54 38 38 73 18 88 88 72 72 64 64 30 35 17 17 98 98 38 38 7 7 74 74 46 46 57 57 5 43 79 79 28 28 72 72 89 53 39 39 78 78 56 56 18 10 20 20 20 20 19 19 58 58 74 74 12 12 65 65 46 66 30 30 93 93 41 41 52 86 18 18 93 93 53 53 56 56 24 24 40 40 36 36 49 49 50 50 35 35 53 53 28 9 99 99 74 74 19 19 64 97 80 80 14 14 98 98 26 17 16 16 59 59 69 69 46

ПРИЛОЖЕНИЕ Б. ИСХОДНЫЙ КОД ПРОГРАММЫ

main.cpp:

```
#include <iostream>
#include <cctype>
#include <cstdlib>

using namespace std;

int *vector_modify(int* vector,int start,int end){
   int j=0;
   for(int i=start;i<end;i++, j++){
      vector[j]=vector[i];
   }
   return vector;
}

void divide_vector(int* vector,int size,int *myVector,
                                                int* pointer,int k){
   k++;
   for(int j=0; j<k;j++){
      cout << '\t';
   }
   for(int i=0;i<size;i++)
      std::cout << vector[i] << " ";
   cout << '\n';
   if(size <= 2){
      for(int i=0;i<size;i++){
            myVector[*(pointer)]=vector[i];
            (*(pointer))++;
      }
   }
   else{
      if((size % 2) == 0){
         divide_vector(vector_modify(vector,0,(size/2)),
                                                size/2,myVector,pointer,k);
         divide_vector(vector_modify(vector,(size/2),size),
                                                size/2,myVector,pointer,k);
      }else{
         divide_vector(vector_modify(vector,0,(size+1)/2),
                                                (size+1)/2,myVector,pointer,k);
```

```
            divide_vector(vector_modify(vector,((size+1)/2)-1,size),
                                                    (size+1)/2,myVector,pointer,k);
        }
    }
}

int main(){
    int num=0;
    while(num != 3){
        cout << "Выберите дальнейшие дествия и введите цифру:\n";
        cout << "1. Ввести координаты вектора вручную.\n";
        cout << "2. Сгенерировать координаты вектора.\n";
        cout << "3. Завершить работу.\n";
        cin >> num;
        switch(num){
            case 1:{
                string c;
                int j=0, k=-1,space=0;
                getchar();
                getline(cin, c);
                for(char i : c){
                    if(isspace(i))
                        space++;
                }
                space++;
                int * vector = new int[space];
                for(char i : c){
                    if(isdigit(i) != 0){
                        vector[j]=int(i)-'0';
                        j++;
                    }
                }
                if(j < 1){
                    cout << "Проверьте введенные
                                                данные и повторите попытку." << endl;
                }else{
                int *myVector = new int[space*2];
                int pointer=0;
                divide_vector(vector,j,myVector,&pointer,0);
                cout << "Результат:";
                for(int i=0;i<pointer;i++)
                    std::cout << myVector[i] << " ";
```

```
    cout << endl;
    delete [] vector;
    delete [] myVector;
    }
    break;
}
case 2:{
    int size = 0, from = 0, to = 0;
    cout << "Введите промежуток через пробел,
    в котором будет сгенерирована длина вектора и его значения: ";
    cin >> from;
    cin >> to;
    if(from > to)
        cout << "Проверьте введенные данные
                                        и повторите попытку." << endl;
    else{
        size = from + rand() % to;
        int* vector = new int[size];
        cout << "Размер вектора: " << size << endl;
        for(int i=0;i<size; i++)
            vector[i] = from + rand() % to;
        cout << "Был сгенерирован вектор:\n";
        for(int i=0;i<size; i++)
            cout << vector[i] << ' ';
        cout << '\n';
        int *myVector = new int[size*2];
        int pointer=0;
        divide_vector(vector,size,myVector,&pointer,0);
        cout << "Результат:";
        for(int i=0;i<pointer;i++)
            std::cout << myVector[i] << " ";
        cout << endl;
        delete [] vector;
        delete [] myVector;
    }
    break;

}
case 3:
    return 0;
default:
```

```
            cout << "Проверьте введенные
                                                данные и повторите попытку." << endl;
            break;
        }

    }
    return 0;
}
```