

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: «Рекурсия»

Студент гр. 7381 _____

Преподаватель _____

Тарасенко Е. А.

Фирсов М. А.

Санкт-Петербург

2018

Задание.

18. Функция Φ преобразования целочисленного вектора α определена следующим образом:

$$\Phi(\alpha) = \begin{cases} \alpha, & \text{если } \|\alpha\| = 1, \\ ab, & \text{если } \|\alpha\| = 2, \alpha = ab, \text{ и } a \leq b, \\ ba, & \text{если } \|\alpha\| = 2, \alpha = ab, \text{ и } b < a, \\ \Phi(\beta)\Phi(\gamma), & \text{если } \|\alpha\| > 2, \alpha = \beta\gamma, \text{ где } \|\beta\| = \|\gamma\|, \|\beta\| = \|\gamma\| + 1. \end{cases}$$

Например: $\Phi(1,2,3,4,5) = 1,2,3,4,5$; $\Phi(4,3,2,1) = 3,4,1,2$; $\Phi(4,3,2) = 3,4,2$.

Отметим, что функция Φ преобразует вектор, не меняя его длину. Реализовать функцию Φ рекурсивно.

Пояснение задания.

На вход программе подаётся последовательность чисел – координат вектора. Задача состоит в преобразовании этих координат по определённому выше правилу, используя рекурсивный алгоритм.

Описание алгоритма.

Основная работа алгоритма заключается в проверке выполнения условий задачи, т. е. если у вектора одна координата, то он записывается в ответ в неизменном виде, тоже самое происходит и с двумерным вектором (если вторая координата не меньше первой, в противном случае в ответе координаты поменяются местами). Если вектор имеет более двух координат, то работа выполняется для первых двух, а оставшиеся передаются в следующий «шаг» рекурсии.

Функции и структуры данных.

void function(int* vector, int* result, int count, int n) – рекурсивная функция, получающая на вход указатель на исходную последовательность координат (int* vector), указатель на

последовательность, которая должна стать результирующей (`int* result`), `count` – число координат и `n` – число, которое будет являться счетчиком «шагов» рекурсии. На каждом «шаге» рекурсии функция выводит на экран результат уже проделанной работы с отступами, соответствующими глубине рекурсии.

int main() – главная функция программы, в которой сначала происходит объявление и инициализация переменных, необходимых для работы. В этом блоке происходит стартовое выделение памяти (4 байта, размер одной переменной типа «`int`») под последовательность координат начального вектора при помощи функции библиотеки языка Си `<stdlib.h> malloc()`.

После ввода первого элемента (координаты), программа принимает символы-разграничители (пробелы) и следующие координаты вектора, попутно перевыделяя память под указатель на него с помощью функции **realloc()**.

Вводимые элементы проверяются на корректность, т. к. принимаются они как последовательности символов. Позже они проходят проверку при помощи функции библиотеки `<stdlib.h> atoi()`, которая возвращает число, если переданная ей строка с него начинается или состоит из него и `NULL`, если нет. В случае, если в качестве координаты программе была передана строка, программа выводит предупреждение об ошибке с указанием ошибочного элемента.

После окончания ввода программа отводит под результирующий указатель столько же памяти, сколько потребовалось в итоге стартовому. Затем вызывается рекурсивная функция **function()**, которая заполняет

результатирующий массив, согласно условия работы. В конце происходит вывод ответа и очистка памяти при помощи функции **free()**.

Тестирование.

Таблица 1 - Примеры условий и ответов

Исходное выражение:	Результат:
1	1
1 2	1 2
4 3 2	3 4 2
1 2 3 4 5	1 2 3 4 5
4 3 2 1	3 4 1 2
12 33 45 g	Wrong coordinate (g)!
hey 3 4 56	Wrong coordinate (hey)!

Вывод.

В ходе данной лабораторной работы были приобретены необходимые навыки по использованию рекурсивных алгоритмов. На языке Си была составлена программа по преобразованию вектора, без изменения его длины.

Приложение 1. Код программы.

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#define N 100

// recursive function;
void function(int* vector, int* result, int count, int n)
{
    for(int i = 0; i < n; i++) printf("\t");
    printf("%d step of the recursion: ", n);
    if(count == 1){
        result[0] = vector[0];
        printf("%d\n", result[0]);
    }
    if(count >= 2){
        if(vector[1] >= vector[0]){
            result[0] = vector[0];
            result[1] = vector[1];
        }
        else{
            result[0] = vector[1];
            result[1] = vector[0];
        }
        printf("%d %d\n", result[0], result[1]);
        // recursion;
        if(count > 2)
            function(vector + 2, result + 2, count - 2, n + 1);
    }
    for(int i = 0; i < n; i++)
        printf("\t");
    printf("end of %d rec. step.\n", n);
}

int main()
{
    int* vector = (int*)malloc(sizeof(int)); // start vect.;
    int* result; // result vector;
    int i = 1;
    char s[N];
    int n = 1; // rec. step;

    // Entering the data;
    printf("Please, enter start vector coordinates: ");
    scanf("%s", s);
    if(atoi(s))
        vector[i-1] = atoi(s);
    else{
        printf("\x1b[31mWrong coordinate (%s)!\n\x1b[0m", s);
        free(vector);
        return 0;
    }
}
```

```

while(getchar() != '\n')
{
    i++;
    vector = realloc(vector, i * sizeof(int));
    s[0] = '\0';
    scanf("%s", s);
    if(atoi(s))
        vector[i - 1] = atoi(s);
    else{
        printf("\x1b[31mWrong coordinate (%s)!\n\x1b[0m", s);
        free(vector);
        return 0;
    }
}

// Proccessing program;
result = (int*)malloc(i*sizeof(int));
printf("\n");
function(vector, result, i, n);
printf("\n");

// Print result and free memory;
printf("Result vector: ");
for(int j = 0; j < i; j++)
    printf("%d ", result[j]);
printf("\n");
free(vector);
free(result);
return 0;
}

```