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. 8381 _____ **Переверзев Д.Е.**

Преподаватель _____ **Жангиров Т.Р.**

Санкт-Петербург
2019

Цель работы

Ознакомиться с основными методами использования рекурсии и написать программу с использованием рекурсии.

Теоретические положения

Рекурсия — определение, описание, изображение какого-либо объекта или процесса внутри самого этого объекта или процесса, то есть ситуация, когда объект является частью самого себя.

Задание

Вариант №19

Функция Ф преобразования целочисленного вектора определена следующим образом:

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

Например:

Ф(1,2,3,4,5) = 1,2,2,3,3,4,4,5;

Ф(1,2,3,4,5,6) = 1,2,2,3,4,5,5,6;

Ф(1,2,3,4,5,6,7) = 1,2,3,4,4,5,6,7;

Ф(1,2,3,4,5,6,7,8) = 1,2,3,4,5,6,7,8.

Отметим, что функция Ф может удлинять вектор. Реализовать функцию Ф рекурсивно.

Выполнение работы

1. Создание массива целочисленных значений(*int* v*)
2. Реализация ввода данных тремя способами:
   - Из консоли
   - Из файла
   - Из аргументов программы

   Если входные данные передаются в аргументы программы, то происходит перезапись значений в созданный массив (v)

   Если данные не были переданы в аргументы программы , то нужно выбрать другой вариант ввода, а именно ввести в консоль:
   - file - если нужно считать из файла;
   - console - если нужно считать из консоли.
3. Создание вспомогательных функций:
   - input() / input_f() - считывание из консоли/файла:

     Считывание чисел происходит до переноса строки

     В аргументы функции передаются ссылка на массив ссылка на длину массива
   - add() - выделение дополнительной памяти для массива:

     В аргументы передаются массив, ссылка на длину массива и число, на которое нужно увеличить массив
   - output() - вывод массива:

     В аргументы переедают массив и его длину
4. Создание основной рекурсивной функции - fun()

   В аргументы принимает массив и ссылку на его длину
   - Если длина меньше 3, то функция возвращает массив
   - Если больше 2, то создается два вспомогательных массива, в которые передаются две половины массива, если длина нечетная , то средний символ передается в оба массива. Затем эти два массива обрабатываются с помощью рекурсивного вызова функции fun() с данными этих массивов. Далее два массива последовательно записываются в один и функция возвращает этот массив.

Выводы

В ходе лабораторной работы были изучены основы работы с рекурсивными функциями. А также реализована рекурсивный вызов функции.

Тестирование программы

```
134:labs_part_1 Dmitriy$ ./a.out
consolee
Wrong word.
```

```
134:labs_part_1 Dmitriy$ ./a.out 1 2 3 4 5
1 2 2 3 3 4 4 5
```

```
134:labs_part_1 Dmitriy$ ./a.out
file
1 2 2 3 4 5 5 6
```

```
134:labs_part_1 Dmitriy$ ./a.out
console
1 2 3 4 5 6 7
1 2 3 4 4 5 6 7
```

```
134:labs_part_1 Dmitriy$ ./a.out 1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
```

```
134:labs_part_1 Dmitriy$ ./a.out
bla bla bla
Wrong word.
```

---

* По умолчанию в файле хранится стока «1 2 3 4 5 6».

Приложение А. Исходный код программы.

```
#include <iostream>
#include <fstream>
#include <string>
#define FILE "./input.txt"
using namespace std;

void input(int *&v, int &len_v);
void input_f(int *&v, int &len_v);
int *add(int *arr, int &len, int k);
void output(int *v, int len_v);
int *fun(int *v, int &len_v);
ifstream in(FILE);

int main(int argc, char* argv[])
{
  int len_v = 1;
  int *v = new int[argc-1];
  if(argc>1)
  {
            len_v=argc-1;
            for(int i=0;i<argc-1;i++)
                      v[i]=atoi(argv[i+1]);
  }
  else
  {
            string str;
            getline(cin,str);
            if(str=="file")
            {
                      string file;
                      input_f(v,len_v);
            }
            else if(str=="console")
                      input(v, len_v);
            else
            {
                      cout<<"Wrong word.\n";
                      return 0;
            }
  }
  v = fun(v, len_v);
  output(v, len_v);
  delete[] v;
  return 0;
}

int *add(int *arr, int &len, int k)
{
```

```
  int *dop = new int[len + k];
  memcpy(dop, arr, len * sizeof(int));
  len += k;
  delete[] arr;
  return dop;
}

void input(int *&v, int &len_v)
{
  char cbuf;
  int i = 0;
  while (true)
  {
            cin >> v[i++];
            cin.get(cbuf);
            if (cbuf == '\n')
                      return;
            v = add(v, len_v, 1);
  }
}

void input_f(int *&v, int &len_v)
{
  char cbuf;
  int i = 0;
  while (true)
  {
            in >> v[i++];
            in.get(cbuf);
            if (cbuf == '\n')
                      return;
            v = add(v, len_v, 1);
  }
}

void output(int *v, int len_v)
{
  for (int i = 0; i < len_v; i++)
            cout << v[i] << " ";
  cout << endl;
}

int *fun(int *v, int &len_v)
{
  if(len_v>2)
  {
            int n= len_v % 2;
            int len_a = len_v / 2 + n;
            int *a = new int[len_a];
            int len_b = len_v / 2 + n;
            int *b = new int[len_b];
```

```
            memcpy(a, v, (len_a) * sizeof(int));
            memcpy(b, &v[len_a - n], (len_b) * sizeof(int));
            a = fun(a, len_a);
            b = fun(b, len_b);
            a = add(a, len_a, len_b);
            memcpy(&a[len_a / 2], b, len_b * sizeof(int));
            len_v = len_a;
            return a;
    }
    return v;
}
```