МИНИСТЕРСТВО ТРАНСПОРТА И КОММУНИКАЦИЙ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Белорусский государственный университет транспорта»

Кафедра «Информационно-управляющие системы и технологии»

Отчет по практическим работам  
по дисциплине «Основы алгоритмизации и программирования»

Выполнил Проверил

студент группы ГИ-11 ст. преп. каф. ИУСиТ

Зайцев И. С. Голдобина Т. А.

Гомель, 2020

Оглавление

[Практическая работа №13 АЛГОРИТМЫ СОРТИРОВКИ 3](#_Toc43128655)

[Задания 3](#_Toc43128658)

[Контрольные вопросы 15](#_Toc43128687)

[Практическая работа №14 Алгоритмы поиска 17](#_Toc43128689)

[Задания 17](#_Toc43128692)

[Контрольные вопросы 23](#_Toc43128718)

[Практическая работа № 19 Тема: «Разработка алгоритма, составление, отладка и выполнение программы с использванием рекурсивных функций» 25](#_Toc43128720)

[Задания 25](#_Toc43128722)

[Контрольные вопросы 30](#_Toc43128737)

[Практическая работа № 21 Тема: «Разработка, отладка и выполнение программы обработки файлов» 31](#_Toc43128739)

[Задания 31](#_Toc43128741)

[Контрольные вопросы 41](#_Toc43128770)

[Практическая работа № 22 Тема: «Разработка, отладка и выполнение программы обработки файлов в С++» 45](#_Toc43128772)

[Задания 45](#_Toc43128774)

[Контрольные вопросы 56](#_Toc43128799)

# Практическая работа №13 АЛГОРИТМЫ СОРТИРОВКИ

Цель

Разработать алгоритмы сортировки.

## Задания

**Условие**

Разработать проект, включающий главную функцию, содержащую меню, и функции-подпрограммы:

1. Ввода или генерации одномерного массива. Количество элементов - не менее 10. Диапазон - на выбор. Вариативность оценивается (тип и диапазон значений массива, количество элементов).
2. Функция, позволяющая поменять местами значения двух чисел (не включается в меню) swap(a, b)
3. Функция сортировки пузырьком
4. Функция сортировки выбором
5. Функция сортировки вставками
6. Функция быстрой сортировки
7. Функция пирамидальной сортировки
8. Функция шейкер-сортировки
9. Функция сортировки Шелла
10. Функция сортировки слиянием
11. Функция сортировки подсчетом
12. Другие функции сортировки (3)
13. Функция вывода одномерного массива

Задание 1.3

**Условие**

Функция сортировки пузырьком

**Программный код**

void bubble(int \*num, int size) // Сортировка пузырьком

{

for (int i=0; i<size-1; i++)

{

for (int j=(size-1); j>i; j--)

{

if (num[j-1]>num[j])

{

int temp=num[j-1];

num[j-1]=num[j];

num[j]=temp;

}

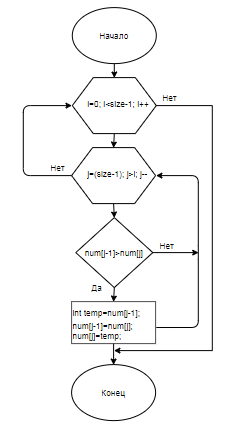
}

}

}

  
Рисунок 1 – Решение задания 1.3

**Блок-схема:**



### **Задание 1.4**

### Условие

Функция сортировки выбором

**Программный код**

void selection(int \*num, int size) // Сортировка выбором

{

int min, temp;

for (int i=0; i<size-1; i++)

{

min = i;

for (int j=i+1; j<size; j++)

{

if (num[j]<num[min])

min=j;

temp=num[i];

num[i]=num[min];

num[min]=temp;

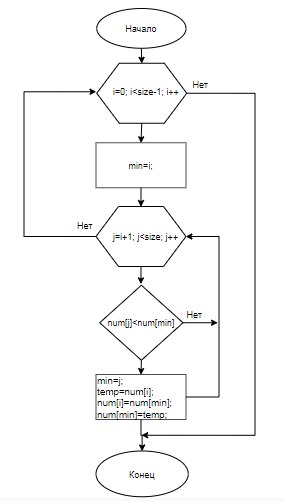
}

}

}

  
Рисунок 1 – Решение задания 1.4

**Блок-схема:**



### **Задание 1.5**

### Условие

Функция сортировки вставками

### Программный код

void insertsort(int \*num, int size) // Сортировка вставками

{

int i, j, x;

for (i=0; i<size; i++)

{

x = num[i];

for (j=i-1; j>=0 && num[j]>x; j--) num[j+1] = num[j];

num[j+1] = x;

}

}

  
Рисунок 1 – Решение задания 1.5

**Блок-схема:**

### 

### **Задание 1.6**

### Условие

Функция быстрой сортировки

### Программный код

void quickSort(int \*numbers, int left, int right)

// Быстрая сортировка

{

int pivot;

int l\_hold = left;

int r\_hold = right;

pivot = numbers[left];

while (left < right)

{

while ((numbers[right] >= pivot) && (left < right))

right--;

if (left != right)

{

numbers[left] = numbers[right];

left++;

}

while ((numbers[left] <= pivot) && (left < right))

left++;

if (left != right)

{

numbers[right] = numbers[left];

right--;

}

}

numbers[left] = pivot;

pivot = left;

left = l\_hold;

right = r\_hold;

if (left < pivot)

quickSort(numbers, left, pivot - 1);

if (right > pivot)

quickSort(numbers, pivot + 1, right);

}

  
Рисунок 1 – Решение задания 1.6

### **Задание 1.7**

### Условие

Функция пирамидальной сортировки

### Программный код

void siftDown(int \*numbers, int root, int bottom) // Пирамидальная сортировка

{

int maxChild;

int done = 0;

while ((root \* 2 <= bottom) && (!done))

{

if (root \* 2 == bottom)

maxChild = root \* 2;

else if (numbers[root \* 2] > numbers[root \* 2 + 1])

maxChild = root \* 2;

else

maxChild = root \* 2 + 1;

if (numbers[root] < numbers[maxChild])

{

int temp = numbers[root];

numbers[root] = numbers[maxChild];

numbers[maxChild] = temp;

root = maxChild;

}

else

done = 1;

}

}

void heapSort(int \*numbers, int array\_size) // Пирамидальная сортировка

{

for (int i = (array\_size / 2) - 1; i >= 0; i--)

siftDown(numbers, i, array\_size - 1);

for (int i = array\_size - 1; i >= 1; i--)

{

int temp = numbers[0];

numbers[0] = numbers[i];

numbers[i] = temp;

siftDown(numbers, 0, i - 1);

}

}

  
Рисунок 1 – Решение задания 1.7

**Блок-схема:**

### 

### **Задание 1.8**

### Условие

Функция шейкер-сортировки

### Программный код

void shaker(int \*num, int size) // Шейкер сортировка

{

int tmp = 0, high = 0, low = (10 - 1);

while (high < low)

{

for (int i = high; i < low; i++)

{

if (num[i] > num[i + 1])

{

tmp = num[i];

num[i] = num[i + 1];

num[i + 1] = tmp;

}

}

low--;

for (int j = low; j > high; j--)

{

if (num[j] < num[j - 1])

{

tmp = num[j];

num[j] = num[j - 1];

num[j - 1] = tmp;

}

}

high++;

}

}

  
Рисунок 1 – Решение задания 1.8

**Блок-схема:**

### 

### **Задание 1.9**

### Условие

Функция сортировки Шелла

### Программный код

void shellSort(int \*num, int size) // Сортировка Шелла

{

int increment = 3;

while (increment > 0)

{

for (int i=0; i<10; i++)

{

int j = i;

int temp = num[i];

while ((j >= increment) && (num[j - increment] > temp))

{

num[j] = num[j - increment];

j = j - increment;

}

num[j] = temp;

}

if (increment > 1)

increment = increment / 2;

else if (increment == 1)

break;

}

}

  
Рисунок 1 – Решение задания 1.9

**Блок-схема:**

### 

### **Задание 1.10**

### Условие

Функция сортировки слиянием

### Программный код

void mergeSort(int \*num, int size) // Сортировка слиянием

{

int step = 1;

int \*temp = (int\*)malloc(size \* sizeof(temp));

while (step < size)

{

int index = 0;

int l = 0;

int m = l + step;

int r = l + step \* 2;

{

m = m < size ? m : size;

r = r < size ? r : size;

int i1 = l, i2 = m;

for (; i1 < m && i2 < r; )

{

if (num[i1] < num[i2]) { temp[index++] = num[i1++]; }

else { temp[index++] = num[i2++]; }

}

while (i1 < m) temp[index++] = num[i1++];

while (i2 < r) temp[index++] = num[i2++];

l += step \* 2;

m += step \* 2;

r += step \* 2;

} while (l < size);

for (int i = 0; i < size; i++)

num[i] = temp[i];

step \*= 2;

}

}

  
Рисунок 1 – Решение задания 1.10

**Блок-схема:**

### 

### **Задание 1.13**

### Условие

Функция вывода одномерного массива

**Программный код**

void display(int num[11]) // Вывод одномерного массива

{

int i;

for (i=0; i<10; i++) {

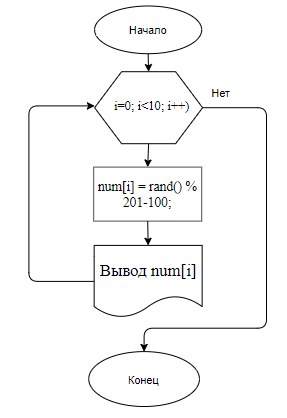
num[i] = rand() % 201-100;

printf("%d ", num[i]);

}}

  
Рисунок 1 – Решение задания 1.13

**Блок-схема:**



## Контрольные вопросы

1. Показатели эффективности алгоритмов – время выполнения программы и требуемый объём памяти.

От чего зависит время выполнения алгоритмов?

1. От сложности алгоритмов, от количества обрабатываемых данных.
2. Асимптотический анализ сложности алгоритмов

Это Оценка времени исполнения алгоритмов. О(n) - n операций, О(n²) - n² операций

1. Сложность алгоритма и понятие О-нотации

Запись вида f(n) = O(g(n)) означает, что ф-ия f(n) возрастает медленнее чем ф-ия g(n) при с = с1 и n = N, где c1 и N могут быть сколь угодно большими числами, т.е. При c = c1 и n >= N, c\*g(n) >=f(n).

О-означает верхнее ограничение сложности алгоритма.

1. Классы сложности

f(n) = O(1) константа  
f(n) = O(log(n)) логарифмический рост  
f(n) = O(n) линейный рост  
f(n) = O(n\*log(n)) квазилинейный рост  
f(n) = O(n^m) полиномиальный рост  
f(n) = O(2^n) экспоненциальный рост

1. Описание алгоритма сортировки выбором

В неотсортированном массиве ищется локальный максимум (минимум).

Найденный максимум (минимум) меняется местами с последним (первым) элементом в массиве.

Если в массиве остались неотсортированные элементы — повторяем действия.

1. На 6-10. Описание алгоритма сортировки быстрой

Из массива выбирается некоторый опорный элемент a[i],

запускается процедура разделения массива, которая перемещает все ключи, меньшие, либо равные a[i], влево от него, а все ключи, большие, либо равные a[i] - вправо,

теперь массив состоит из двух подмножеств, причем левое меньше, либо равно правого,

для обоих подмассивов: если в подмассиве более двух элементов, рекурсивно запускаем для него ту же процедуру.

1. На 8-10. Описание алгоритма сортировки шейкер

От последней перестановки до конца (начала) массива находятся отсортированные элементы. Учитывая данный факт, просмотр осуществляется не до конца (начала) массива, а до конкретной позиции. Границы сортируемой части массива сдвигаются на 1 позицию на каждой итерации.

Массив просматривается поочередно справа налево и слева направо.

Просмотр массива осуществляется до тех пор, пока все элементы не встанут в порядке возрастания (убывания).

Количество просмотров элементов массива определяется моментом упорядочивания его элементов.

1. На 9-10. Описание алгоритма сортировки слиянием

Разделим массив пополам, рекурсивно отсортируем части, после чего выполним процедуру слияния: поддерживаем два указателя, один на текущий элемент первой части, второй – на текущий элемент второй части. Из этих двух элементов выбираем минимальный, вставляем в ответ и сдвигаем указатель, соответствующий минимуму.

1. На 10. Описание алгоритма сортировки на выбор (отличной от предыдущих)

Сортировка пузырьком:

Будем идти по массиву слева направо. Если текущий элемент больше следующего, меняем их местами. Делаем так, пока массив не будет отсортирован. Заметим, что после первой итерации самый большой элемент будет находиться в конце массива, на правильном месте. После двух итераций на правильном месте будут стоять два наибольших элемента, и так далее.

Вывод по работе

Разработал алгоритмы сортировки.

# Практическая работа №14 Алгоритмы поиска

Цель

Разработать алгоритмы поиска.

## Задания

### Условие

Разработать проект, включающий главную функцию, содержащую меню, и функции-подпрограммы:

1. Ввода или генерации одномерного массива. Количество элементов - не менее 10. Диапазон - на выбор. Вариативность оценивается (тип и диапазон значений массива, количество элементов).
2. Функция, позволяющая поменять местами значения двух чисел (не включается в меню) swap(a, b)
3. Функция сортировки (на выбор)
4. Функция последовательного прямого поиска
5. Функция бинарного поиска
6. Функция поиска методом транспозиции
7. Функция поиска с перемещением в начало
8. Функция индексно-последовательного поиска
9. Функция вывода одномерного массива

### **Задание 1.1**

### Условие

Ввода или генерации одномерного массива. Количество элементов - не менее 10. Диапазон - на выбор. Вариативность оценивается (тип и диапазон значений массива, количество элементов).

### Листинг

void out(int Mass[10], int size)

{

for(int i=0; i<size; i++)

{

printf("%d ", Mass[i]);

}

printf("\n");}

### **Задание 1.3**

### Условие

Функция сортировки (на выбор).

### Листинг

void bubbleSort(int Mass[10], int size)

{for (int i=0; i<10-1; i++)

{

for (int j=(size-1); j>i; j--)

{

if (Mass[j-1]>Mass[j])

{

swap(&Mass[j], &Mass[j-1]);

}

}

}

}

### **Задание 1.4**

### Условие

Функция последовательного прямого поиска

### Листинг

int simpleSearch (int Mass[10], int num[10], int size, int value)

{

int j=0;

for (int i=0; i<size; i++)

{

if (Mass[i]==value)

{

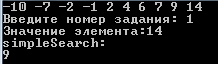
num[j++]=i;

}

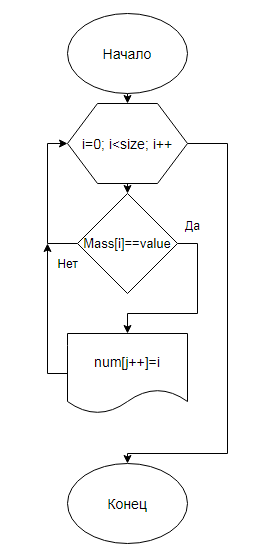
}

return j;

}

  
Рисунок 1 – Решение задания 1.4

**Блок-схема:**



### **Задание 1.5**

### Условие

Функция бинарного поиска

### Листинг

int binarySearch(int Mass[10], int left, int right, int value)

{

if (left<right)

{

int medium=(left+right)/2;

if (Mass[medium]<value)

{

return binarySearch(Mass, medium+1, right, value);}

else if (Mass[medium]>value)

{

return binarySearch(Mass, left, medium-1, value);}

else

{return medium;}

} else if (left==right && Mass[left]==value)

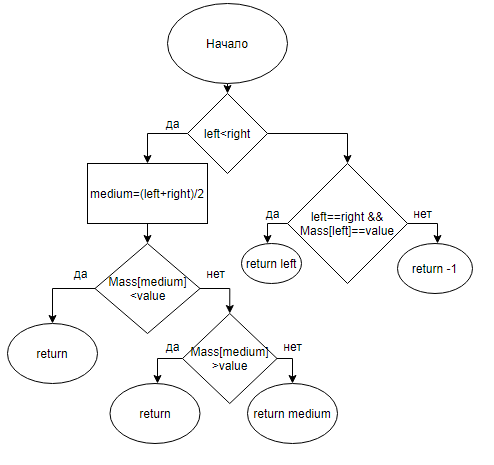
{return left;}

else {return -1;}

}

  
Рисунок 1 – Решение задания 1.5

**Блок-схема:**



### **Задание 1.6**

### Условие

Функция поиска методом транспозиции

### Листинг

int transposSearch(int Mass[10], int size, int value)

{

if (Mass[0]==value){return 0;}

else

{for (int i=1; i<size;i++)

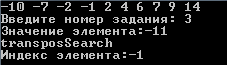
{if (Mass[i]==value){

swap(&Mass[i],&Mass[i-1]);

return i;}}}

return -1;

}

  
Рисунок 1 – Решение задания 1.6

### **Задание 1.7**

### Условие

Функция поиска с перемещением в начало

### Листинг

int searchNear(int Mass[10], int size, int pos)

{

if (pos!=-1)

{

int j=1;

int temp=pos-1;

while (temp>=0 && Mass[temp]==Mass[pos]){

temp--;

j++;}

temp=pos+1;

while (temp<size && Mass[temp]==Mass[pos]){

temp++;

j++;}

return j;

} else {return 0;}

}

int startSearch(int Mass[10], int size, int value)

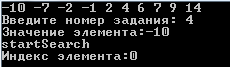
{

for (int i=0; i<size;i++){if (Mass[i]==value){

swap(&Mass[i],&Mass[0]);

return i;}}

return -1;}

  
Рисунок 1 – Решение задания 1.7

### **Задание 1.8**

### Условие

Функция индексно-последовательного поиска

### Листинг

int indexSearch(int Mass[10], int size, int value)

{

int indexTable[size/3], i;

for (i=2; i<size; i+=3)

{indexTable[i/3]=Mass[i];}

for (i=0; i<size/3; i++){

if(indexTable[i]>value){

for(int j=i\*3; j<(i+1)\*3; j++)

{if (Mass[j]==value){return j;}}

return -1;}

else if (indexTable[i]==value)

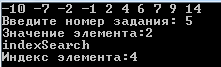
{return i\*3+2;}}

for (i=(i+1)\*3; i<size; i++)

{if (indexTable[i]==value){return i;}}

return -1;

}

  
Рисунок 1 – Решение задания 1.8

### **Задание 1.9**

### Условие

Функция вывода одномерного массива

### Листинг

void out(int Mass[10], int size)

{

for(int i=0; i<size; i++)

{

printf("%d ", Mass[i]);

}

printf("\n");}

## Контрольные вопросы

1. Описание алгоритма последовательного прямого поиска

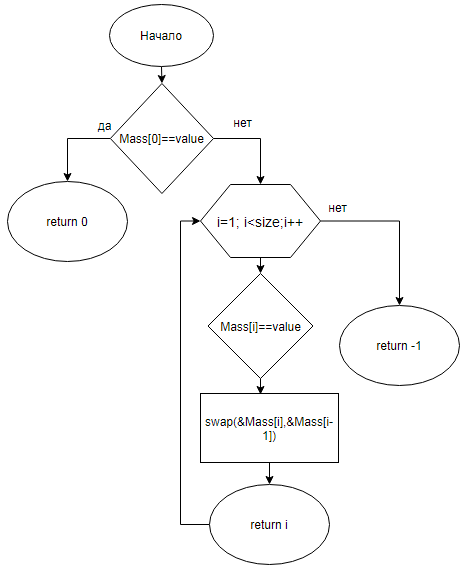
Идём по порядку по массиву, находим нужный элемент.

1. Описание алгоритма бинарного поиска

Двоичный поиск заключается в том, что на каждом шаге множество объектов делится на две части и в работе остаётся та часть множества, где находится искомый объект..

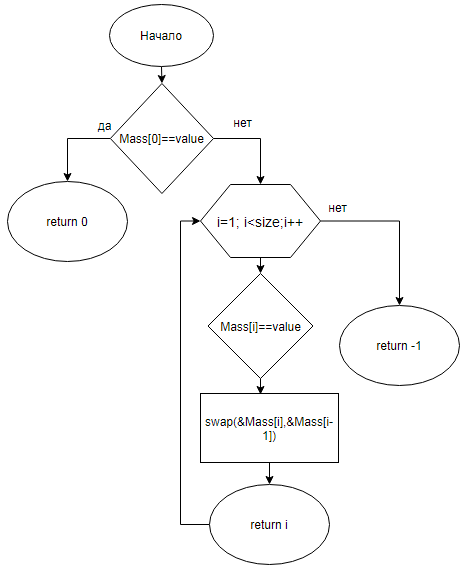
1. На 7-10. Описание алгоритма последовательного поиска с транспозицией. Блок-схема алгоритма

Последовательный поиск с транспозицией это улучшенная версия последовательного поиска. Каждый запрос к элементу сопровождается сменой мест этого и предшествующего элемента. В итоге наиболее часто используемые элементы постепенно перемещаются в начало массива. И при последующем обращении к ним, эти записи находятся почти сразу.



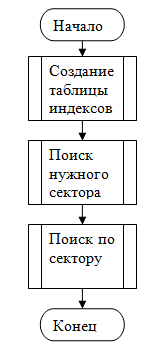
1. На 7-10. Описание алгоритма последовательного поиска с перемещением в начало. Блок-схема алгоритма

В этом методе каждый запрос к элементу сопровождается его перемещением в начало массива. В итоге в начале массива оказывается элемент, используемый в последний раз.



1. На 7-10. Описание алгоритма индексно-последовательного поиска. Блок-схема алгоритма

Достоинство алгоритма индексно-последовательного поиска заключается в том, что сокращается время поиска, так как последовательный поиск первоначально ведется в индексной таблице, имеющей меньший размер, чем основная таблица. Когда найден правильный индекс, второй последовательный поиск выполняется по небольшой части записей основной таблицы.



Вывод по работе

Разработал алгоритмы поиска.

# Практическая работа № 19 Тема: «Разработка алгоритма, составление, отладка и выполнение программы с использванием рекурсивных функций»

### **Цель**

Разработать алгоритмы, составить и выполнить программы с использованием рекурсивных функций.

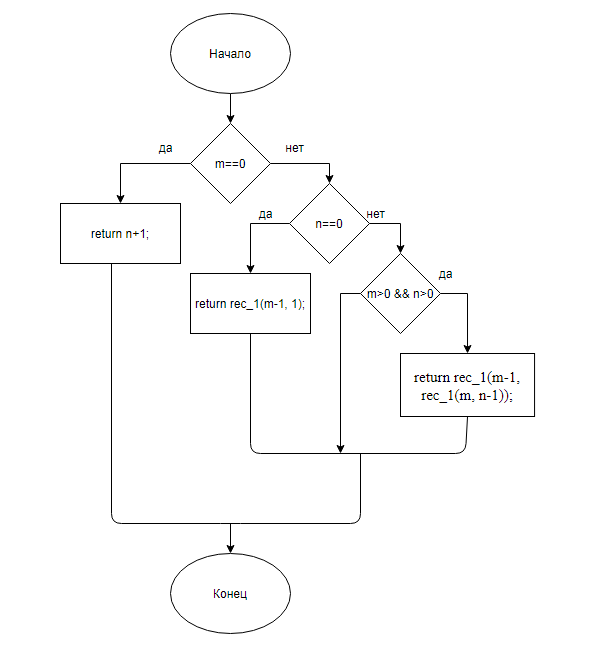
## Задания

### Задание 1

### Условие

Решить задачу двумя способами – с применением рекурсии и без нее.Найти значение функции Аккермана A(m, n), которая определяется для всех неотрицательных целых аргументов m и n следующим образом: A(0, n)= n + 1,если m = 0; A(m,0) = A(m – 1, 1), если n = 0; A(m, n) = A(m – 1, A(m, n – 1)), если и m > 0, и n > 0.

**Блок-схема**



### Программный код

void task\_1(int m, int n)

{int key;

printf("Рекурсия - %d\n", key=rec\_1(m, n));

printf("Проверка через цикл - %d\n", key);

}

int rec\_1(int m, int n)

{

if(m==0)

return n+1;

else if(n==0)

return rec\_1(m-1, 1);

else if(m>0 && n>0)

return rec\_1(m-1, rec\_1(m, n-1));

}

### Результат

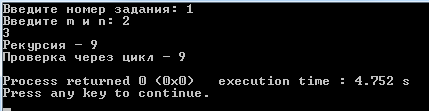


Рисунок – Решение задания 1

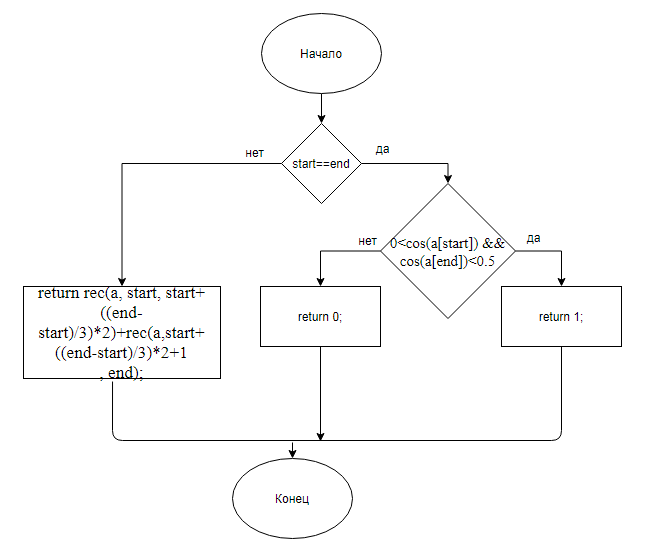
### Задание 2

### Условие

### Ввести с клавиатуры (сгенерировать) одномерный массив целых чисел. Решить задачу путем рекурсивного разбиения массива на две части. Для контроля решить задачу с использованием циклического алгоритма.

Определить количество элементов массива, для которых выполняется условие 0 < cos(a[i]) < 0.5. При рекурсивном разбиении массив делить на первые 2/3 и остальную треть массива.

### Блок-схема



### Программный код

void task\_2(int \*a)

{int key=0, start=0,end=9, i;

for(i=0; i<10; i++)

{

if(0<cos(a[i]) && cos(a[i])<0.5)

{

key++;

}

}

printf("\nРекурсия - %d\n", rec(a, start, end));

printf("Проверка через цикл - %d\n", key);

}

int rec(int \*a, int start, int end)

{

if(start==end)

{

if(0<cos(a[start]) && cos(a[end])<0.5)

{

return 1;

}

else {return 0;}

}

else

{

return rec(a, start, start+((end-start)/3)\*2)+rec(a,start+((end-start)/3)\*2+1 , end);

}

}

### Результат

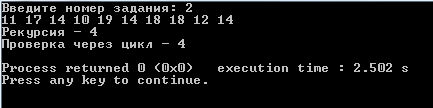


Рисунок 2 – Решение задания 2

### Задание 3

### Условие

Вычислить сумму ряда из задания 3 практической работы № 9 с применением рекурсии.

### Программный код

double rec\_3(int n,double x, double eps, double Sn)

{

if(fabs(Sn)<eps)

return 0;

Sn\*=(-1)\*2\*x/n;

++n;

return Sn + rec\_3(n, x, eps, Sn);

}

void task\_3()

{

int n=1;

double x, eps, S, Sn;

printf("x=");

scanf("%lf", &x);

while(!(x>=-2 && x<=-1))

{printf("x=");

scanf("%lf",&x);

}

printf("eps=");

scanf("%lf",&eps);

S=Sn=1;

S += rec\_3(n,x,eps,Sn);

printf("S= %.15lf\n", S);

printf("e^(-2x)= %lf",exp(-2\*x));

}

### Результат

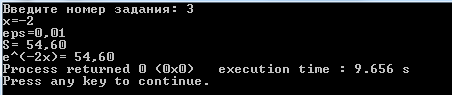


Рисунок 3 – Решение задания 3

## Контрольные вопросы

1. Понятие рекурсии.

Рекурсия – это функция, которая сама вызывает себя.

1. Структура рекурсивной функции.

Рекурсивная функция состоит из двух частей: базовой и рекурсивной.

1. Условие окончания рекурсивного алгоритма.

В тексте рекурсивной функции обязательно должен быть выполнен условный оператор, например if, который при определенных условиях вызовет завершение функции, т.е. возврат, а не выполнит очередной рекурсивный вызов. Если такого оператора нет, то после вызова функция никогда не сможет завершить работы. Распространенной ошибкой при написании рекурсивных функций как раз и является отсутствие в них условного оператора.

1. Достоинство рекурсии.

Естественность (натуральность) представления сложных, на первый взгляд, алгоритмов;

рекурсивный алгоритм более читабелен в сравнении с итерационным;

для многих распространенных задач рекурсию более легко реализовать чем

итерацию. Рекурсия хорошо подходит для реализации алгоритмов обхода

списков, деревьев, анализаторов выражений, комбинаторных задач и т.д.

1. Недостатки рекурсии.

По сравнению с итерацией многоразовый вызов рекурсивной функции требует больше времени. Это связано с тем, что при вызове рекурсивного метода его параметры копируются в стек. При завершении вызова рекурсивной функции предыдущие значения параметров вытягиваются из стека, что приводит к лишним операциям. Итерационный алгоритм для той же задачи работает быстрее;

### 

### Вывод по практической работе

Разработал алгоритмы, составил и выполнил программы с использованием рекурсивных функций.

# Практическая работа № 21 Тема: «Разработка, отладка и выполнение программы обработки файлов»

### **Цель**

Разработать, отладить и выполнить программы для обработки файлов.

## Задания

### Задание 1

### Условие

Составить функцию, с помощью которой открыть текстовый файл, записать в него символ и закрыть файл. Открыть тот же файл для чтения, считать символ и вывести в консоль.  
Выполнить тесты для цифр, латинских букв верхнего и нижнего регистров и кириллических букв верхнего и нижнего регистров.

### Листинг

void task\_1()

{ int arr[N];

FILE \*fp;

fp=fopen("file.txt", "w");

fputc('a', fp);

fclose(fp);

fp=fopen("file.txt", "r");

while (fgets (arr, N, fp) != NULL)

printf("%s", arr);

printf("\n");

fclose(fp);

}

### Результат

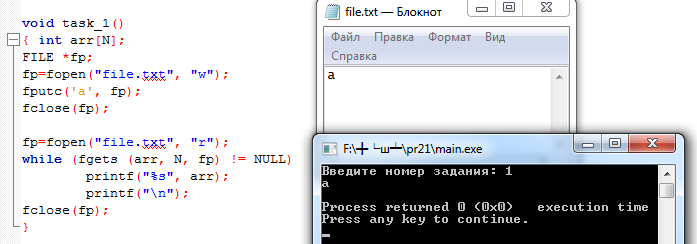


Рисунок – Решение задания 1

### Задание 2

### Условие

Составить функцию, с помощью которой открыть текстовый файл, записать в него строку, массив чисел и закрыть файл. Далее открыть тот же файл, считать информацию и вывести в консоль.

### Листинг

void task\_2()

{

int i,n, arr[N];

char a[20];

int b[] = {1, 2, 5};

FILE \*file1;

scanf("%s", a);

file1 = fopen("file.txt", "w");

fprintf(file1, "%s", a);

for(i=0; i<3; i++)

fprintf(file1, " %d", b[i]);

fclose(file1);

file1=fopen("file.txt", "r");

while (fgets (arr, N, file1) != NULL)

printf("%s", arr);

printf("\n");

fclose(file1);

}

### Результат

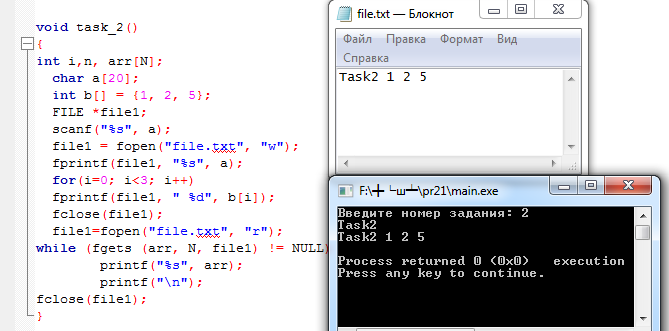


Рисунок 2 – Решение задания 2

### Задание 3

### Условие

Создать файл (вручную). Записать в этот файл число, символы (латинские и кириллица), строки (латинскими и кириллицей). Сохранить, закрыть. Составить функцию, с помощью которой созданный файл открыть, все считать, вывести в консоль и закрыть.

### Листинг

void task\_3()

{ char arr[N];

FILE \*file1;

file1=fopen("task\_3.txt", "r");

while (fgets (arr, N, file1) != NULL)

printf("%s", arr);

printf("\n");

fclose(file1);

}

### Результат

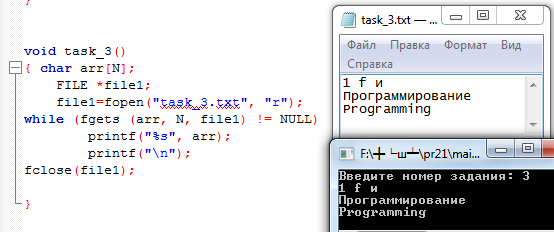


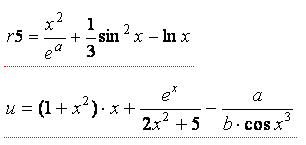
Рисунок 3 – Решение задания 3

### Задание 4

### Условие

Преобразовать программу, разработанную ранее (ПР № 5, Задание 1) таким образом, чтобы входные данные считывались из внешних файлов, а выходные - записывались во внешний файл. Условие задания привести полностью, включая расчетные формулы.

ПР №5, Задание 1:



### Листинг

void task\_4()

{ int x, a, b;

float r5, u;

FILE \*inFile;

FILE \*outFile;

inFile=fopen("input4.txt", "r");

fscanf(inFile, "%d", &x);

fscanf(inFile, "%d", &a);

fscanf(inFile, "%d", &b);

fclose(inFile);

printf("Данные из файла: %d %d %d\n", x,a,b);

r5=((x\*x)/exp(a))+ 1/3.\*sin(x)\*sin(x)-log(x);

u=(1+x\*x)\*x+ (exp(x)/(2\*x\*x+5))- (a/(b\*cos(x)\*cos(x)\*cos(x)));

printf("Ответы: r5=%f, u=%f", r5, u);

outFile=fopen("output4.txt", "w");

fprintf(outFile, "%f\n", r5);

fprintf(outFile, "%f", u);

fclose(outFile);

printf("\n");

}

### Результат

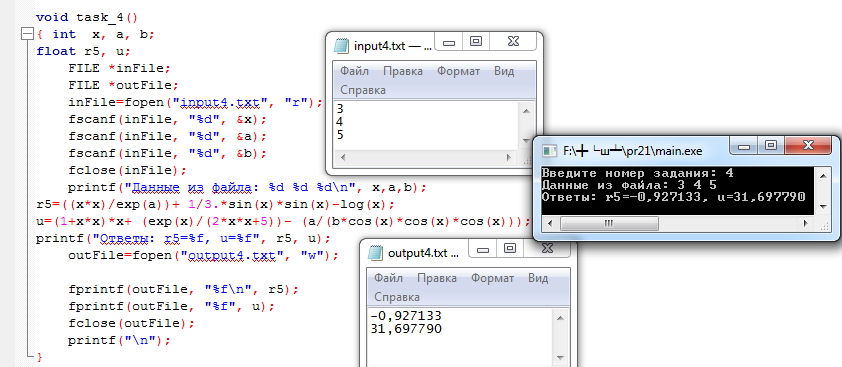


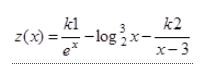
Рисунок 4 – Решение задания 4

### Задание 5

### Условие

Преобразовать программу, разработанную ранее (ПР № 8, Задание 1) таким образом, чтобы входные данные считывались из внешних файлов, а выходные - записывались во внешний файл. Условие задания привести полностью, включая расчетные формулы.

ПР №8, Задание 1:



при каждом значении переменной x Є [1.5 ; 6] , изменяющейся с шагом  Δx=0.3.

Использовать цикл **for**. Значения констант k1 и k2 - произвольные. Учесть возможность возникновения исключений во время выполнения программы.

### Листинг

void task\_5()

{ FILE \*inFile;

FILE \*outFile;

inFile=fopen("input5.txt", "r");

int k1, k2, i=0;

float z,x ;

fscanf(inFile, "%d", &k1);

fscanf(inFile, "%d", &k2);

printf("Входные данные: %d %d\n", k1, k2);

outFile=fopen("output5.txt", "w");

printf("x\t\tz(x)\n");

fclose(inFile);

fprintf(outFile, "x\t\tz(x)\n");

for(x=1.5; x<=6.3; x+=0.3){

i++;

if(i==6)

{

printf("%.3f\t\t--\n", x);

fprintf(outFile, "%.3f\t\t--\n", x);

continue;

}

z=k1\*1./exp(x) - pow(log2(x\*1.),3) - k2\*1./(x-3);

printf("%.3f\t\t%.3f\n", x, z);

fprintf(outFile, "%.3f\t\t%.3f\n", x ,z);}

fclose(outFile);}

### Результат

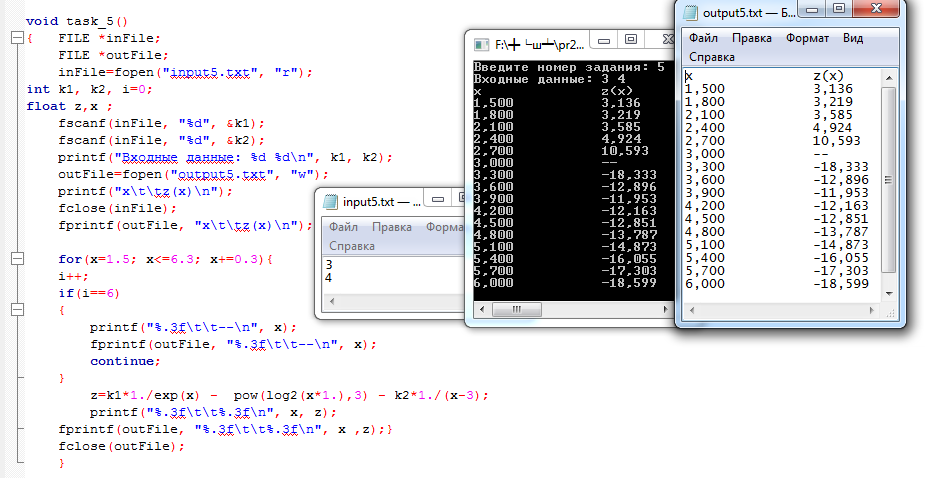


Рисунок 5 - Решение задания 5

### Задание 6

### Условие

Преобразовать программу, разработанную ранее (ПР № 11, Задание 1) таким образом, чтобы входные данные считывались из внешних файлов, а выходные - записывались во внешний файл. Условие задания привести полностью, включая расчетные формулы.

ПР № 11, Задание 1:

Ввести одномерный массив *U* из десяти элементов и число *n*. Для каждого элемента массива вычислить . Вывести полученный массив. Определить произведение отрицательных элементов массива *D*. Поменять местами значения последнего и минимального элементов *D*.

Вывод результатов в процессе отладки программы осуществлять на экран поэлементно в цикле.

### Листинг

void task\_6()

{ FILE \*inFile;

FILE \*outFile;

inFile=fopen("input6.txt", "r");

int U[10], min=0, zam=0;

int i, D[10], n,x=1,z=0;

fscanf (inFile, "%d", &n);

for (i=0; i<10; i++)

{

fscanf(inFile, "%d", &U[i]);

}

fclose(inFile);

outFile=fopen("output6.txt", "w");

for (i=0; i<10; i++)

{D[i]=(1/tan(U[i]))/(n+1)-log(abs(n\*U[i]));}

for (i=0; i<10; i++)

{

fprintf(outFile,"D[%d]=%d\n",i, D[i]);

} for (i=0; i<10; i++) {

if (D[i]<0)

{

x\*=D[i];

}

} fprintf (outFile, "Произведение отрицательных элементов массива: %d\n\n",x);

for (i=0; i<10; i++){

if (min>=D[i]) {min=D[i]; z=i;}

}

zam=D[z];

D[z]=D[9];

D[9]=zam;

fprintf(outFile, "Замена минимального и последнего элемента массива:\n");

for (i=0; i<10; i++)

{

fprintf(outFile,"D[%d]=%d\n",i, D[i]);

}

fclose(outFile);

}

### Результат

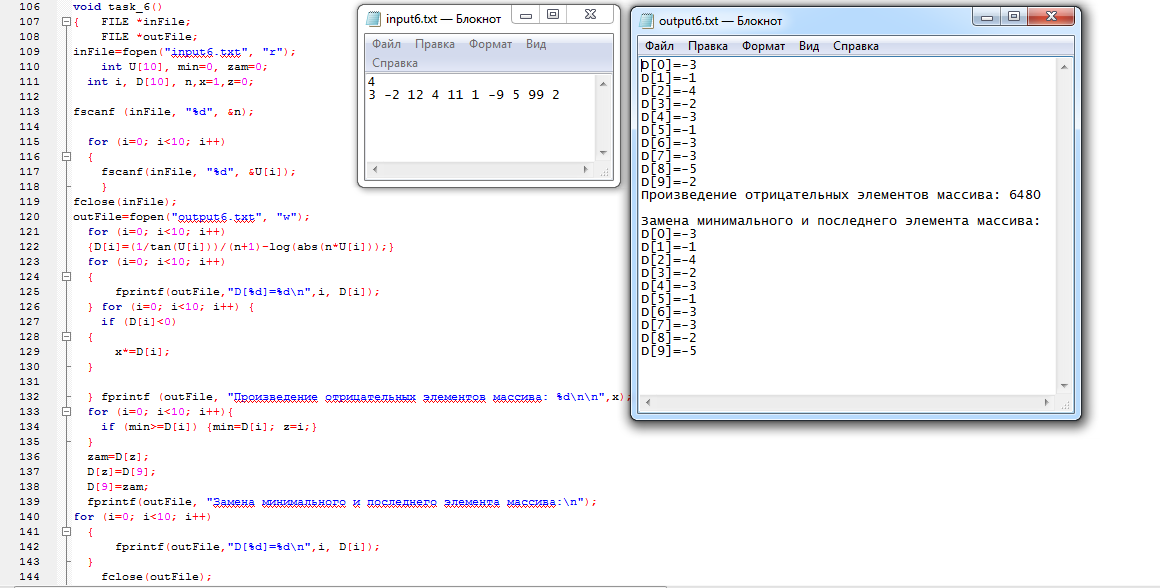


Рисунок 6 - Решение задания 6

### Задание 7

### Условие

Преобразовать программы, разработанные ранее (ПР № 15, Задания 1, 2) таким образом, чтобы входные данные считывались из внешних файлов, а выходные - записывались во внешний файл. Условия заданий привести полностью.

ПР № 15, Задание 1:

Составить программу, в которой вводится строка (не менее 20 символов, осмысленная, содержащая строчные и прописные буквы и цифры, которые могут быть изменены по заданию).

Выводится копия строки, в которой: сделать первые буквы каждого слова строчными, а остальные - прописными, заменить букву "к" на "т" (с учетом регистра).

ПР № 15, Задание 2:

Составить программу, в которой вводится строка (не менее 20 символов, осмысленная, содержащая строчные и прописные буквы и цифры).

Выводится строка, в которой: каждая третья буква, кроме 9-й, заменяется на прописную.

### Листинг

void task\_7\_1()

{FILE \*fp;

int i;

char string[1024], \*mass\_char;

mass\_char=string;

fp=fopen("input7\_1.txt", "r");

fgets(string, 80, fp);

fclose(fp);

printf("%s", string);

for(i=0; i<strlen(string); i++ ){

if(\*mass\_char == 107){\*mass\_char = 116;}

else if(\*mass\_char == 75){\*mass\_char = 84;}

if(\*mass\_char>=65 && \*mass\_char<=90){

\*mass\_char+=32;

fprintf(stdout, "%c", \*mass\_char);}

else if(\*mass\_char==32){

fprintf(stdout, "%c", \*mass\_char);

mass\_char++;

\*mass\_char=tolower(\*mass\_char);

if(\*mass\_char == 107){\*mass\_char = 116;}

fprintf(stdout, "%c", \*mass\_char);

i++;}

else if(\*mass\_char>=97 && \*mass\_char<=122){

\*mass\_char-=32;

fprintf(stdout, "%c", \*mass\_char);}

else {fprintf(stdout, "%c", \*mass\_char);}

mass\_char++;}

fp=fopen("output7\_1.txt", "w");

fprintf(fp, "%s", string);

fclose(fp);

}

void task\_7\_2()

{FILE \*fp;

char string[1024];

char \*mass\_char, ch;

mass\_char=string;

fp=fopen("input7\_2.txt","r");

fgets(string, "%s", fp);

fclose(fp);

printf("\n\n%s\n", string);

for(int i=2; i<7; i+=3 )

{

if (96<string[i] && string[i]<123 || 159<string[i] && string[i]<176)

{

string[i]-=32;

}

else if (223<string[i] && string[i]<240)

{

string[i]-=80;

}

}

for(int i=11; i<strlen(string); i+=3 )

{

if (96<string[i] && string[i]<123 || 159<string[i] && string[i]<176)

{

string[i]-=32;

}

else if (223<string[i] && string[i]<240)

{

string[i]-=80;

}

}

for(int i=0; i<strlen(string); i++ )

{

printf("%c", string[i]);

}

fp=fopen("output7\_2.txt","w");

fprintf(fp, "%s", string);

fclose(fp);

}

### Результат

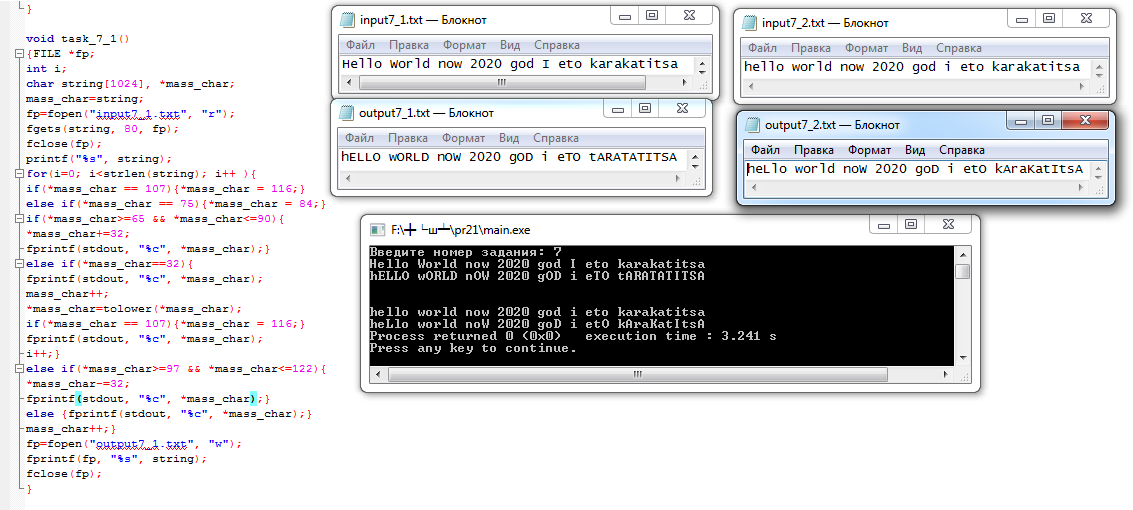


Рисунок 7- Решение задания 7

## 

## Контрольные вопросы

1. Где применяются файлы данных (назначение)?

Работа с файлами, содержащими результаты измерений, в т. ч. полученные автоматически, или отсканированные данные, требующие дальнейшего чтения и анализа. Взаимодействие с другими программами, данные, сохраненные в файлы могут быть импортированы в другие приложения и наоборот. Сохранение информации для повторного использования, сравнения и анализа. Обмен данными и результатами.

1. Классификация файлов данных.

Файлы подразделяют на входные (с исходными данными) и выходными (с результатами работы программы).

1. Этапы работы с файлами данных.

Создать файловый объект (переменную, поток), выполнить чтение / запись файла, закрыть файл (очистить поток).

1. Что такое поток на C/C++?

Файловая система языка С предназначена для работы с самыми разными устройствами, в том числе терминалами, дисководами и накопителями на магнитной ленте. Даже если какое-то устройство сильно отличается от других, буферизованная файловая система все равно представит его в виде логического устройства, которое называется потоком.

1. Виды потоков файловой системе C/C++.

Текстовый и бинарный.

1. Особенности текстового потока.

Текстовый поток — это последовательность символов. В стандарте С считается, что текстовый поток организован в виде строк, каждая из которых заканчивается символом новой строки. Однако в конце последней строки этот символ не является обязательным. В текстовом потоке по требованию базовой среды могут происходить определенные преобразования символов. Например, символ новой строки может быть заменен парой символов — возврата каретки и перевода строки. Поэтому может и не быть однозначного соответствия между символами, которые пишутся (читаются), и теми, которые хранятся во внешнем устройстве. Кроме того, количество тех символов, которые пишутся (читаются), и тех, которые хранятся во внешнем устройстве, может также не совпадать из-за возможных преобразований.

1. Указатель файла. Объявление указателя.

Указатель файла — это то, что соединяет в единое целое всю систему ввода/вывода языка С. Указатель файла — это указатель на структуру типа FILE. Он указывает на структуру, содержащую различные сведения о файле, например, его имя, статус и указатель текущей позиции в начало файла. В сущности, указатель файла определяет конкретный файл и используется соответствующим потоком при выполнении функций ввода/вывода. Чтобы выполнять в файлах операции чтения и записи, программы должны использовать указатели соответствующих файлов. Чтобы объявить переменную-указатель файла, используйте такого рода оператор: FILE \*fp;

1. Функция fopen(). Назначение.

Функция fopen() открывает поток и связывает с этим потоком определенный файл. Затем она возвращает указатель этого файла. Чаще всего (а также в оставшейся части этой главы) под файлом подразумевается дисковый файл. Прототип функции fopen() такой:

FILE \*fopen(const char \**имя\_файла*, const char \**режим*);

где имя\_файла — это указатель на строку символов, представляющую собой допустимое имя файла, в которое также может входить спецификация пути к этому файлу. Строка, на которую указывает *режим*, определяет, каким образом файл будет открыт.

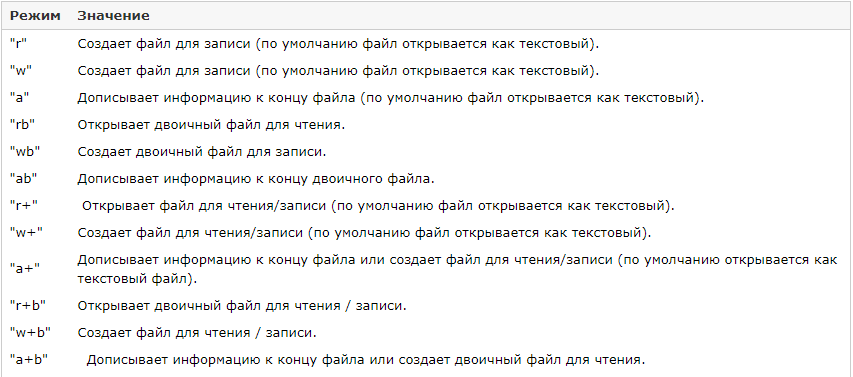
1. Функция fclose(). Назначение.

Функция fclose() закрывает поток, который был открыт с помощью вызова fopen().Функция fclose() записывает в файл все данные, которые еще оставались в дисковом буфере, и проводит, так сказать, официальное закрытие файла на уровне операционной системы. Отказ при закрытии потока влечет всевозможные неприятности, включая потерю данных, испорченные файлы и возможные периодические ошибки в программе. Функция fclose() также освобождает блок управления файлом, связанный с этим потоком, давая возможность использовать этот блок снова. Так как количество одновременно открытых файлов ограничено, то, возможно, придется закрывать один файл, прежде чем открывать другой. Прототип функции fclose() такой:

int fclose(FILE \*уф);

где уф — указатель файла, возвращенный в результате вызова fopen(). Возвращение нуля означает успешную операцию закрытия. В случае же ошибки возвращается EOF. Чтобы точно узнать, в чем причина этой ошибки, можно использовать стандартную функцию ferror() (о которой вскоре пойдет речь). Обычно отказ при выполнении fclose() происходит только тогда, когда диск был преждевременно удален (стерт) с дисковода или на диске не осталось свободного места.

1. Режимы открытия файла.



1. Функции для чтения символов из файлов. Описание.

Для ввода символа также имеются две эквивалентные функции: getc() и fgetc(). Обе определяются для сохранения совместимости со старыми версиями С. В этой книге используется getc() (которая обычно реализуется в виде макроса), но если хотите, применяйте fgetc().

Функция getc() записывает символы в файл, который с помощью fopen() уже открыт в режиме для чтения. Прототип этой функции следующий:

int getc(FILE \*fp);

где fp— это указатель файла, имеющий тип FILE и возвращенный функцией fopen(). Функция getc() возвращает целое значение, но символ находится в младшем байте. Если не произошла ошибка, то старший байт (байты) будет обнулен.

1. Функция ferror(). Назначение и применение.

Функция ferror() определяет, произошла ли ошибка во время выполнения операции с файлом. Прототип этой функции следующий:

int ferror(FILE \*fp);

где fp — допустимый указатель файла. Она возвращает значение true (истина), если при последней операции с файлом произошла ошибка; в противном же случае она возвращает false (ложь). Так как при любой операции с файлом устанавливается свое условие ошибки, то после каждой такой операции следует сразу вызывать ferror(), а иначе данные об ошибке могут быть потеряны.

### Вывод по практической работе

Разработал, отладил и выполнил программы для обработки файлов.

# Практическая работа № 22 Тема: «Разработка, отладка и выполнение программы обработки файлов в С++»

### **Цель**

Разработать, отладить и выполнить программы для обработки файлов в C++.

## Задания

### Задание 1

### Условие

Составить функцию с использование объектов cout и cin, которая запрашивает с клавиатуры 6 вещественных чисел, записывает эти числа в массив, а затем записывает их в текстовый файл tdata.

### Листинг

task1()

{

float Mass[N];

int i;

cout <<"Введите элементы массива(6)"<<endl;

for(i=0; i<N; i++)

{

cin >> Mass[i];

}

ofstream fp("tdata.txt");

if(fp == NULL)

{

cout << "Ошибка при открытии файла" << endl;

return -1;

}

for(i=0; i<N; i++)

{

fp << Mass[i] << " ";

}

fp.close();

cout <<endl;

}

### Результат

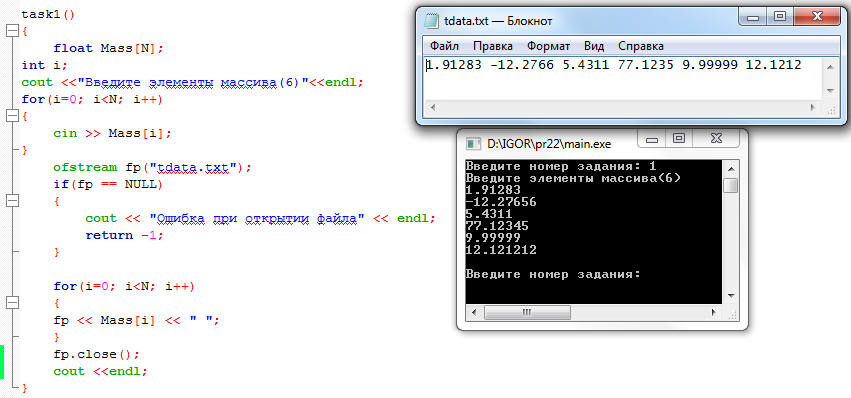


Рисунок – Решение задания 1

### Задание 2

### Условие

Составить функцию, которая считывает числа из текстового файла tdata и печатает их на экране с четырьмя знаками после запятой.

### Листинг

task2()

{

float Mass[N];

int i;

ifstream fp("tdata.txt", ios::binary);

if(fp == NULL)

{

cout << "Ошибка при открытии файла" << endl;

return -1;

}

for(i=0; i<N; i++)

{

fp >> Mass[i];

}

fp.close();

for(i=0; i<N; i++)

{

cout << round(Mass[i]\*10000)/10000 <<" "; //Округление до 4 знаком после запятой

}

cout <<endl;

}

### Результат

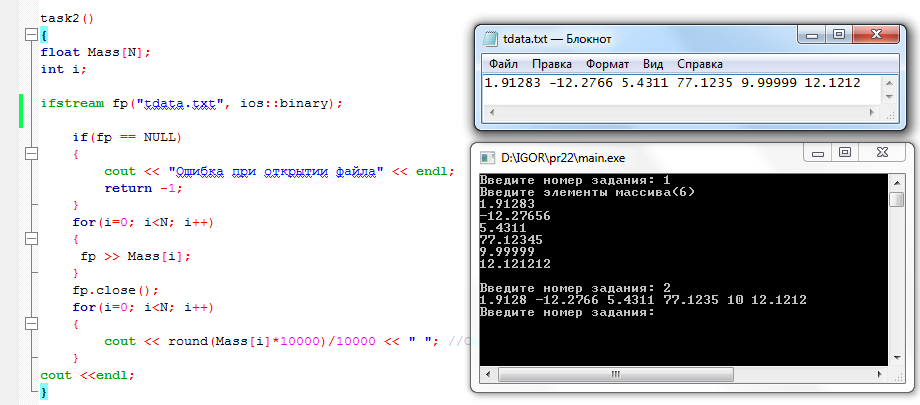


Рисунок 2 – Решение задания 2

### Задание 3

### Условие

Составить функцию, которая генерирует 31 вещественное чис(ла/ел/ло), лежащ(их/ee) в диапазоне от -23 до -4 и помещает их в массив. Записать сгенерированный массив в бинарный файл bdata.

### Листинг

task3()

{

float Mass[31];

int i, max=-4, min=-23;

for(i=0; i<31; i++)

{

Mass[i] = (float) rand()/RAND\_MAX\*(max-min)+min;

if(Mass[i]>-4){Mass[i]-=5;}

cout << Mass[i] <<" ";

}

ofstream fp("bdata.txt", ios::binary);

if(fp == NULL)

{

cout << "Ошибка при открытии файла" << endl;

return -1;

}

for(i=0; i<31; i++)

{

fp << Mass[i] << " ";

}

fp.close();

cout << endl;

}

### Результат

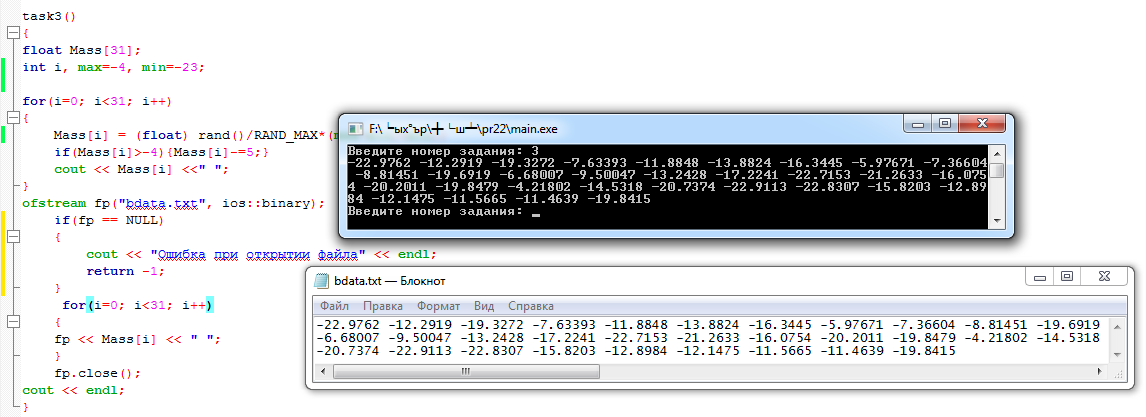


Рисунок 3 – Решение задания 3

### Задание 4

### Условие

Написать программу, которая открывает бинарный файл bdata, печатает все числа из этого файла, печатает максимальное и минимальное из значений этих чисел. Предусмотреть проверку открытия файла.

### Листинг

task4()

{

float Mass[31], max, min;

int i;

ifstream fp("bdata.txt", ios::binary);

if(fp == NULL)

{

cout << "Ошибка при открытии файла" << endl;

return -1;

}

for(i=0; i<31; i++)

{

fp >> Mass[i];

}

fp.close();

max=Mass[0];

min=Mass[0];

for(i=0;i<N; i++)

{

if(Mass[i]>max)

{

max=Mass[i];

}

if(Mass[i]<min)

{

min=Mass[i];

}

}

for(i=0; i<31; i++)

{

cout << Mass[i] <<" ";

}

cout << endl << "Максимальное число:" << max << endl << "Минимальное число:" << min <<endl;

}

### Результат

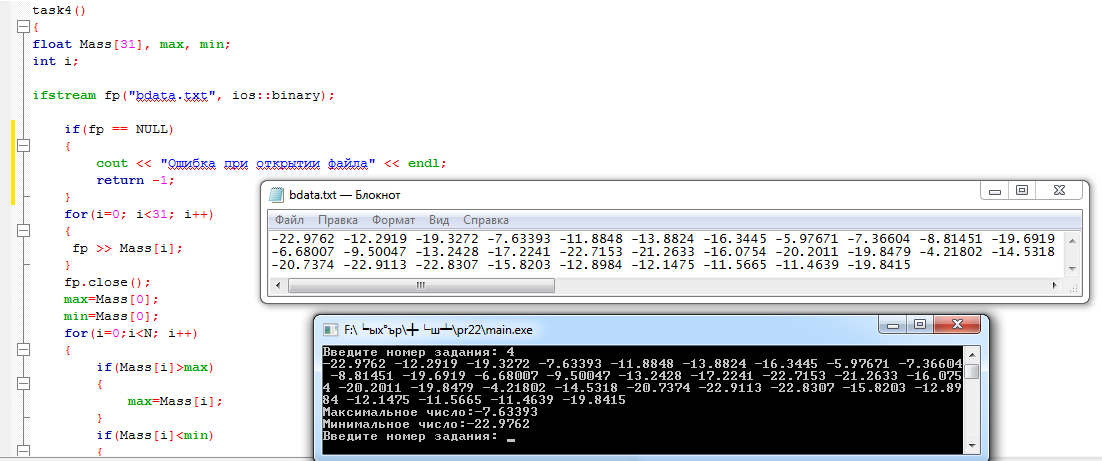


Рисунок 4 – Решение задания 4

### Задание 5

### Условие

Преобразовать программу, разработанную ранее (ПР № 11, Задание 1), заменив функции ввода/вывода языка С на функции языка С++ во всей программе. Условие задания привести полностью, включая расчетные формулы.

ПР № 11, Задание 1:

Ввести одномерный массив *U* из десяти элементов и число *n*. Для каждого элемента массива вычислить . Вывести полученный массив. Определить произведение отрицательных элементов массива *D*. Поменять местами значения последнего и минимального элементов *D*.

Вывод результатов в процессе отладки программы осуществлять на экран поэлементно в цикле.

### Листинг

task5()

{

int U[10], mini, index;

int i, D[10], n, x=1, last, z;

cout << "Введите число n(любое):";

cin >> n;

for (i=0; i<10; i++)

{

cout <<"U["<<i<<"] = ";

cin >> U[i];

}

for (i=0; i<10; i++)

{D[i]=(1/tan(U[i]))/(n+1)-log(abs(n\*U[i]));}

cout << endl;

mini=D[0];

for (i=0; i<10; i++)

{

cout <<"D["<<i<<"]="<<D[i]<<endl;

} for (i=0; i<10; i++) {

if (D[i]<0)

{

x\*=D[i];

}

if (D[i]<mini)

{

mini=D[i];

index=i;

z=D[index];

D[index]=D[9];

D[9]=z;

}

}

cout <<"Произведение элементом массива D<0: "<<x<<endl;

cout <<"После замены минимального и последнего:"<<endl;

for (i=0; i<10; i++)

{

cout <<"D["<<i<<"]="<<D[i]<<endl;

}

}

### Результат

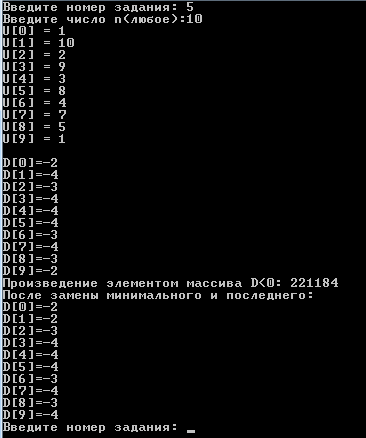


Рисунок 5 - Решение задания 5

### Задание 6

### Условие

Преобразовать программы, разработанные ранее (ПР № 15, Задания 1, 2), заменив функции ввода/вывода языка С на функции языка С++ во всей программе. Условия заданий привести полностью.

ПР № 15, Задание 1:

Составить программу, в которой вводится строка (не менее 20 символов, осмысленная, содержащая строчные и прописные буквы и цифры, которые могут быть изменены по заданию).

Выводится копия строки, в которой: сделать первые буквы каждого слова строчными, а остальные - прописными, заменить букву "к" на "т" (с учетом регистра).

ПР № 15, Задание 2:

Составить программу, в которой вводится строка (не менее 20 символов, осмысленная, содержащая строчные и прописные буквы и цифры).

Выводится строка, в которой: каждая третья буква, кроме 9-й, заменяется на прописную.

### Листинг

task6()

{

char string[1024];

char \*mass\_char;

mass\_char=string;

gets(string); /\*Hello World now 2020 god I eto karakatitsa\*/

for(int i=0; i<strlen(string); i++ ){

if(\*mass\_char == 107){\*mass\_char = 116;}

else if(\*mass\_char == 75){\*mass\_char = 84;}

if(\*mass\_char>=65 && \*mass\_char<=90){

\*mass\_char+=32;

cout << \*mass\_char;}

else if(\*mass\_char==32){

cout << \*mass\_char;

mass\_char++;

\*mass\_char=tolower(\*mass\_char);

if(\*mass\_char == 107){\*mass\_char = 116;}

cout << \*mass\_char;

i++;}

else if(\*mass\_char>=97 && \*mass\_char<=122){

\*mass\_char-=32;

cout << \*mass\_char;}

else {cout << \*mass\_char;}

mass\_char++;}

task7()

{

char string[1024];

char \*mass\_char;

mass\_char=string;

gets(string); /\*hello world now 2020 god i eto karakatitsa\*/

for(int i=2; i<7; i+=3 ){string[i]=toUpperCase(string[i]);}

for(int i=11; i<strlen(string); i+=3 ){string[i]=toUpperCase(string[i]);}

for(int i=0; i<strlen(string); i++ ){cout << string[i];}

}

char toUpperCase (char ch){

if (96<ch && ch<123 || 159<ch && ch<176){return ch-32;}

else if (223<ch && ch<240){return ch-80;}

return ch;

}

### Результат

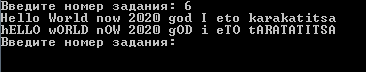


Рисунок 6.1 - Решение задания 6

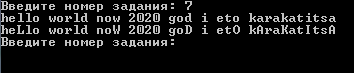


Рисунок 6.2 - Решение задания 6

## 

## Контрольные вопросы

1. Какая библиотека С++ предназначена для потокового ввода/вывода?

#include <iostream>

1. Стандартный поток ввода

cin >>

1. Операции для работы с потоком: вставка в поток

cin >>; cout <<;

1. Манипулятор endl

endl – это манипулятор вывода, который вставляет в выходной поток символ перехода на новую строку, а затем сбрасывает буфер объекта ostream

1. Флаги форматированного ввода/вывода

right – выравнивает по правой границе;

left – выравнивает по левой границе(по умолчанию);

dec – вывод величины в десятичной системе счисления;

showpos – выводит “+” для положительных элементов;

scientific – для вещественных чисел в экспоненциальном формате;

setw(n) – задаёт ширину поля;

setprecision(n) – задает количество цифр в дробной части числа;

1. Вывод вещественных чисел с помощью флагов форматированного вывода

setprecision(n);

1. Определение количества цифр в дробной части числа с помощью функции форматирования

cout.precision(n);

1. Манипуляторы форматирования

setw(n) – задаёт ширину поля;

setprecision(n) – задает количество цифр в дробной части числа;

fixed – фиксированная форма вывода вещественных чисел;

1. Функции защиты от некорректного ввода

-;

1. Классы для работы с файлами

**Класс  ifstream;**

**Класс ofstream;**

1. Работа с текстовыми файлами на уровне потоков

#include <iostream>

#include <fstream>

int main()

{

    std::ofstream out;          // поток для записи

    out.open("D:\\hello.txt"); // окрываем файл для записи

    out.close();            // закрываем файл

    std::ifstream in;       // поток для чтения

    in.open("D:\\hello.txt"); // окрываем файл для чтения

    in.close();             // закрываем файл

    std::fstream fs;        // поток для чтения-записи

    fs.open("D:\\hello.txt"); // окрываем файл для чтения-записи

    fs.close();             // закрываем файл

    return 0;

}

### Вывод по практической работе

Разработал, отладил и выполнил программы для обработки файлов в C++.