Министерство цифрового развития, связи и массовых коммуникаций РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет

телекоммуникаций и информатики» (СибГУТИ)

Кафедра телекоммуникационных сетей и вычислительных средств

**Отчет**

на тему

## **"Задание 13** Практическое задание 13. Анализ трудоемкости методов сортировки **Вариант 3,5"**

Выполнил: студент I курса

ИВТ, гр. ИА-132

Кулаков К.В.

Проверил:

Вейлер А. И

Новосибирск 2022

Вариант 3,5

Команда:

Командир: Кулаков Кирилл

Участники:

Нейдорф Павел

Кузеванов Дмитрий

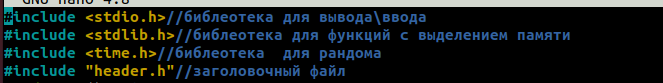
Лазарева Диана

Кацуро Михаил



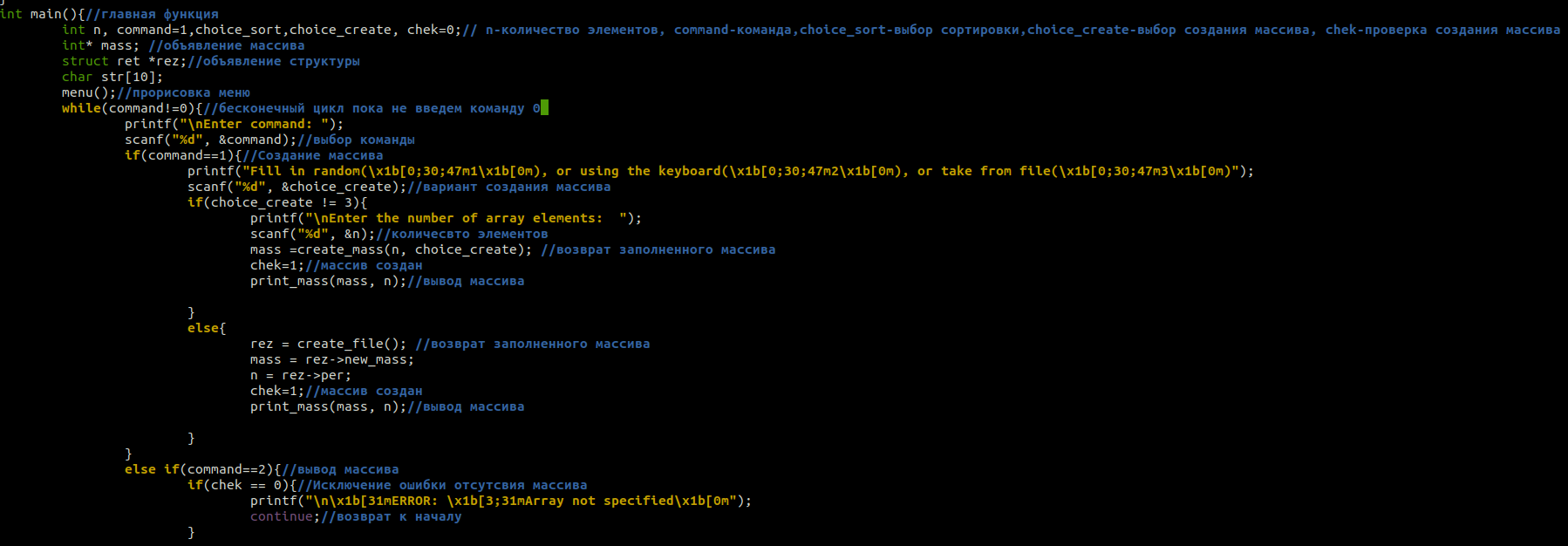


Подключение необходимых библиотек

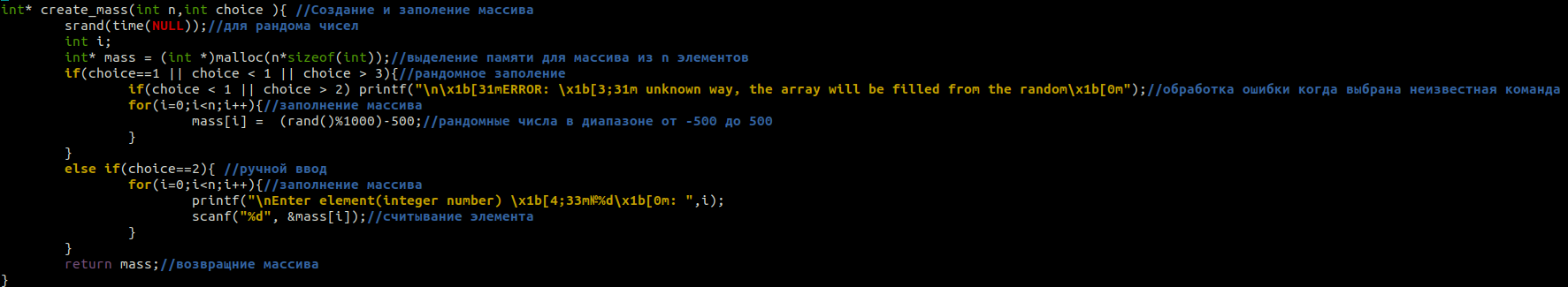


Функция menu:нужна для отрисовки меню программы  

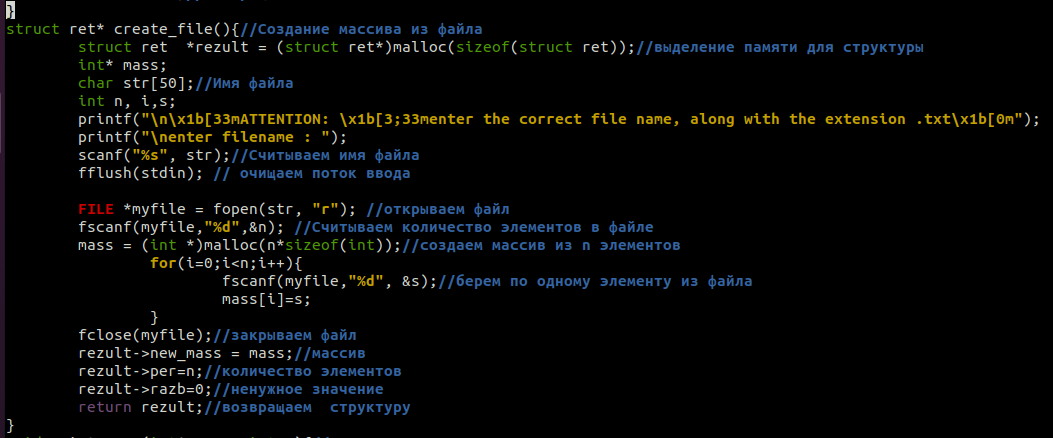

Основная функция main: Здесь происходит выполнение основной программы



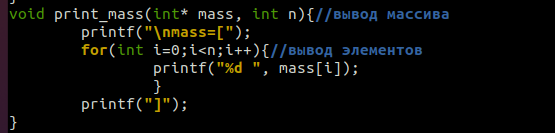
Функция create\_mass: для создания и заполнения массива



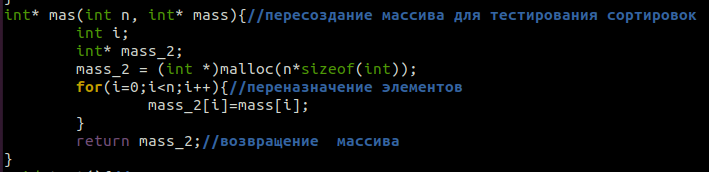
Функция create\_file: Для создания массива из файла



Функция print\_mass: для вывода массива на экран



Функция mas: для пересоздания массива(что бы после сортировки массив был не отсортирован)

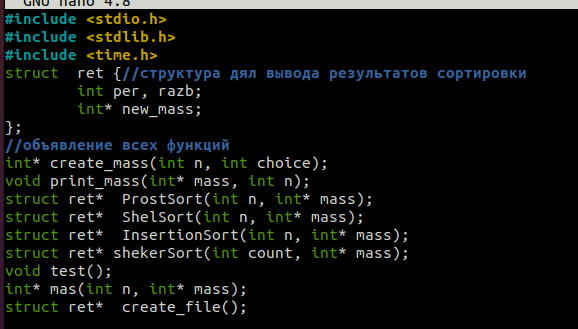


Функция test:для тестирования всех сортировок на трудоемкость

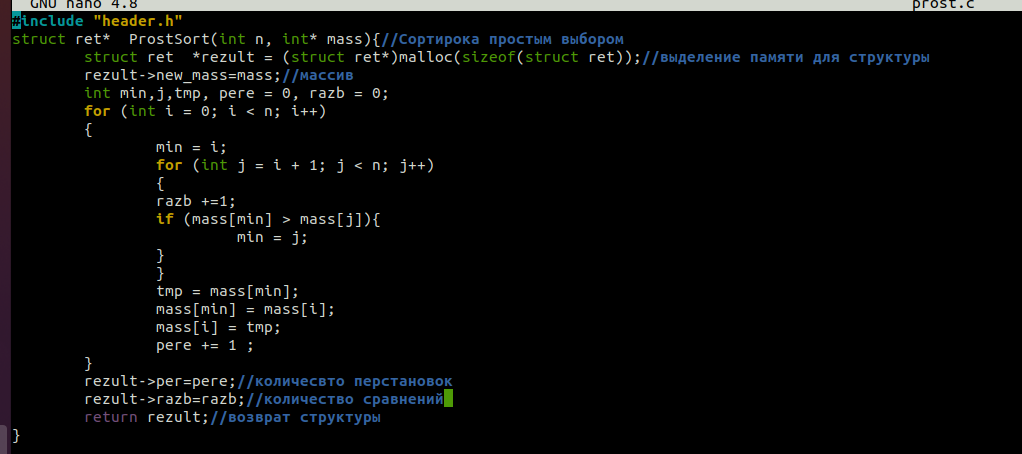




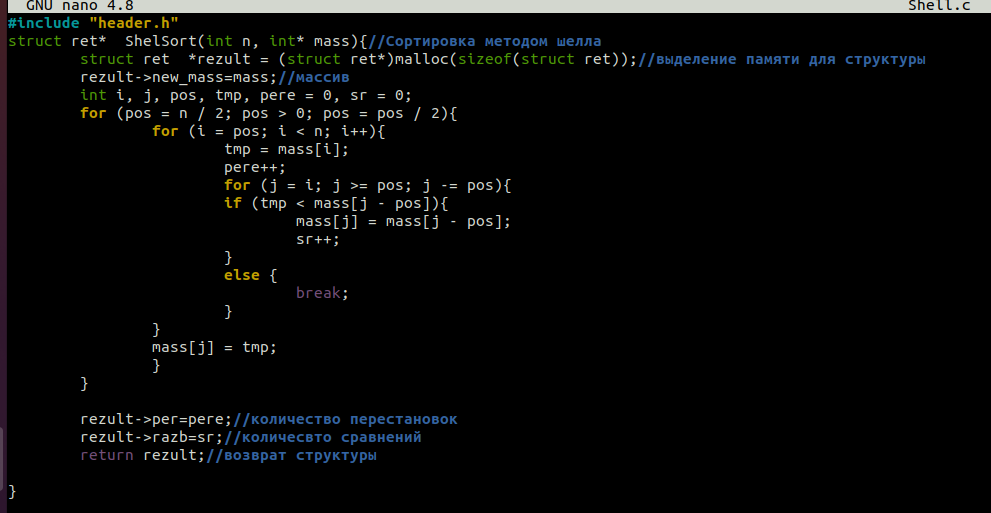
Заголовочный файл



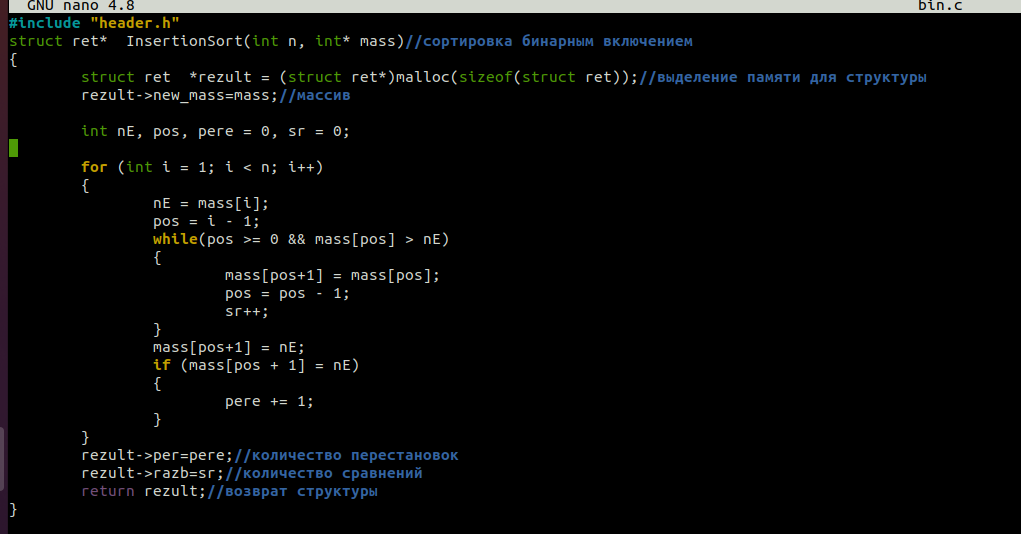
Сортировка простым выбором: Автор Нейдорф Павел



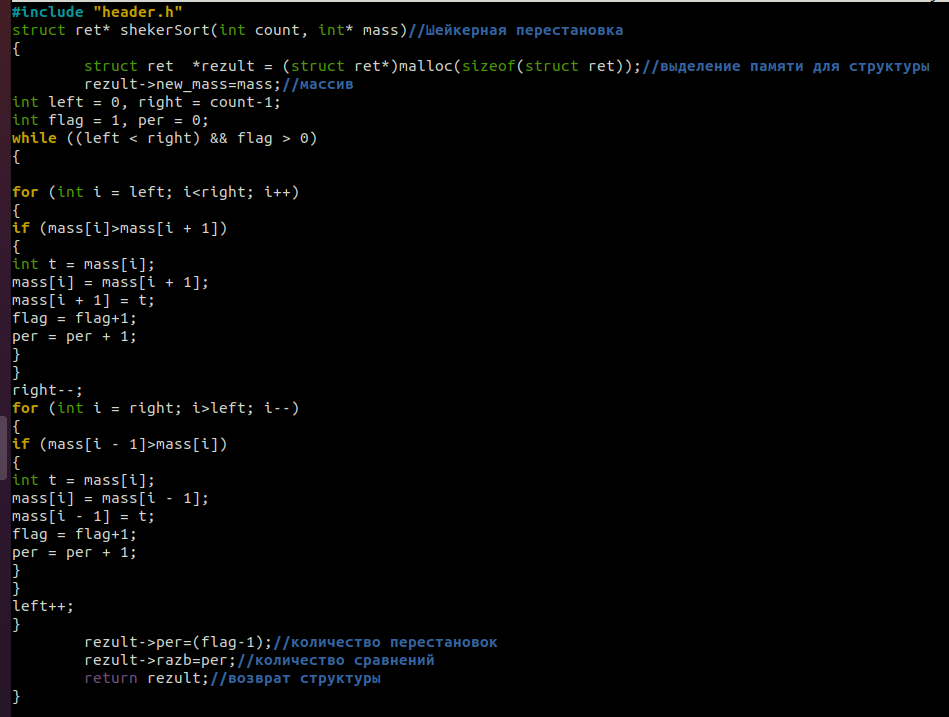
Сортировка методом Шелла: Автор Кузеванов Дмитрий



Сортировка бинарным включением: Автор Лазарева Диана



Шейкерная сортировка: Автор Кацуро Михаил



Анализ трудоемкости сортировок:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Размер массива | Мф+Сф м. Простой выбор | | | Мф+Сф м. Шелла | | |
| Убыв. | Случ. | Возр. | Убыв. | Случ. | Возр. |
| 100 | 5050 | 5050 | 5050 | 534 | 867 | 503 |
| 200 | 20100 | 20100 | 20100 | 1256 | 2237 | 1203 |
| 300 | 45150 | 45150 | 45150 | 2146 | 3783 | 2104 |
| 400 | 80200 | 80200 | 80200 | 2843 | 5724 | 2803 |
| 500 | 125250 | 125250 | 125250 | 3592 | 6771 | 3506 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Размер массива | Мф+Сф м. Бинарное включение | | | Мф+Сф м. Шейкерная сортировка | | |
| Убыв. | Случ. | Возр. | Убыв. | Случ. | Возр. |
| 100 | 771 | 2827 | 99 | 1344 | 5456 | 2500 |
| 200 | 2143 | 9900 | 199 | 3888 | 19402 | 10000 |
| 300 | 2368 | 22097 | 299 | 4138 | 43596 | 22500 |
| 400 | 2778 | 38665 | 399 | 4758 | 76532 | 40000 |
| 500 | 6893 | 63696 | 499 | 12788 | 126394 | 62500 |

Вывод: наилучший способ сортировки – сортировка шелла тк в различных ситуациях сортировка шелла показывает наименьший результат трудоемкости, на втором месте стоит бинарное включение, тк средний показатель трудоемкости немного больше, чем у сортировки Шелла, на третьем месте стоит Шейкерная сортировка тк ее средний показатель трудоемкости гораздо больше, чем у бинарного включения, однако, меньше чем у простого.Наихудшем методом оказался простой выбор, тк в независимости от условий задания массива, трудоемкость оказалась наибольшей

------------------------------------------------------------------------------------

**Файл header.h**

------------------------------------------------------------------------------------

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

struct ret {

int per, razb;

int\* new\_mass;

};

int\* create\_mass(int n, int choice);

void print\_mass(int\* mass, int n);

struct ret\* ProstSort(int n, int\* mass);

struct ret\* ShelSort(int n, int\* mass);

struct ret\* InsertionSort(int n, int\* mass);

struct ret\* shekerSort(int count, int\* mass);

void test();

int\* mas(int n, int\* mass);

struct ret\* create\_file();

------------------------------------------------------------------------------------

**Файл main.c**

------------------------------------------------------------------------------------

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "header.h"

void menu(){

printf("\x1b[0;34m---------------------------------------------------------------------------------------------------------------------");

printf("\n\x1b[0;35m█████████████████████████████████████████████████████████████████████████████████████████████████████████████████████");

printf("\n█ ██ ██ ██ █ ██ ████ ███ █████ ███ ██████ ███ ██ ██ ██ ██ ██ ███ ██ █");

printf("\n█ ██████ ███ ██ ██████ ████████ ████ ██ ████████████████ ██ ████ █████ █ ████████ ██ ██████ █████████ ███ ██ ████");

printf("\n█ ██ ███ ██ ████████ ████ ███ ████ ██ ██ █████ ██ ████ █████ █ █ ██ ██ ██ ██ ██ ███ ██ █");

printf("\n█████ ██ ███ ██ ████████ ████ ███ ████ ██ ████ █████ █████ ██ ████ █████ ███ ██ █████ ██████ ██████ ██ ███ █ ███ ████");

printf("\n█ ██ ██ ████████ ████ ███ ████ ██ █████ ██ ████ █████ ███ ██ ██ ██ ██ ████ ████ █");

printf("\n█████████████████████████████████████████████████████████████████████████████████████████████████████████████████████");

printf("\n \x1b[0;34m---------------------------------------------------------------------------------------------------------------------\x1b[0;37m");

printf("\n \x1b[0;30;47m0\x1b[0m-\x1b[4;33m Exit\x1b[0m \x1b[0;30;47m1\x1b[0m-\x1b[4;33mCreate massive\x1b[0m \x1b[0;30;47m2\x1b[0m-\x1b[4;33m Print massive\x1b[0;37m\x1b[0m ");

printf("\x1b[0;30;47m3\x1b[0m-\x1b[4;33m Sort massive\x1b[0m \x1b[0;30;47m4\x1b[0m-\x1b[4;33m Clear console\x1b[0m ");

}

int main(){

int n, command=1,choice\_sort,choice\_create, chek=0;

int\* mass;

struct ret \*rez;

char str[10];

menu();

while(command!=0){

printf("\nEnter command: ");

scanf("%d", &command);

if(command==1){

printf("Fill in random(\x1b[0;30;47m1\x1b[0m), or using the keyboard(\x1b[0;30;47m2\x1b[0m), or take from file(\x1b[0;30;47m3\x1b[0m)");

scanf("%d", &choice\_create);

if(choice\_create != 3){

printf("\nEnter the number of array elements: ");

scanf("%d", &n);

mass =create\_mass(n, choice\_create);

chek=1;

print\_mass(mass, n);

}

else{

rez = create\_file();

mass = rez->new\_mass;

n = rez->per;

chek=1;

print\_mass(mass, n);

}

}

else if(command==2){

if(chek == 0){

printf("\n\x1b[31mERROR: \x1b[3;31mArray not specified\x1b[0m");

continue;

}

print\_mass(mass, n);

}

else if(command==3){

if(chek == 0){

printf("\n\x1b[31mERROR: \x1b[3;31mArray not specified\x1b[0m");

continue;

}

printf("\nChoose a sorting method\n\x1b[0;30;47m1)\x1b[0m\x1b[4;33m easy choice\n\x1b[0;30;47m2)\x1b[0m\x1b[4;33m shell method");

printf("\n\x1b[0;30;47m3)\x1b[0m\x1b[4;33mbinary inclusion\n\x1b[0;30;47m4)\x1b[0m\x1b[4;33m shaker sort\x1b[0m\n\x1b[0;30;47m5)\x1b[0m\x1b[4;33m testingAll\x1b[0m\n");

printf("Enter your choice: ");

scanf("%d", &choice\_sort);

if (choice\_sort==1){

rez = ProstSort(n, mass);

printf("перестановок-%d, сравнений-%d", rez->per, rez->razb);

print\_mass(mass, n);

}

else if(choice\_sort==2){

rez = ShelSort(n, mass);

printf("перестановок-%d, сравнений-%d", rez->per, rez->razb);

print\_mass(mass, n);

}

else if(choice\_sort==3){

rez = InsertionSort(n, mass);

printf("перестановок-%d, сравнений-%d", rez->per, rez->razb);

print\_mass(mass, n);

}

else if(choice\_sort==4){

rez = shekerSort(n, mass);

printf("перестановок-%d, сравнений-%d", rez->per, rez->razb);

print\_mass(mass, n);

}

else if(choice\_sort==5) {

test();

}

else if(choice\_sort<1|| choice\_sort>5){

printf("\n\x1b[31mERROR: \x1b[3;31mNon-existent sort method -> a easy choice sort will be performed\x1b[0m");

rez = ProstSort(n, mass);

printf("перестановок-%d, разбиений-%d", rez->per, rez->razb);

}

}

else if(command==4){

system("clear");

menu();

}

else if(command != 0) printf("\n\x1b[31mERROR: \x1b[3;31mcommand not found\x1b[0m");

}

}

------------------------------------------------------------------------------------

**Файл main\_2.c**

#include "header.h"

int\* create\_mass(int n,int choice ){

srand(time(NULL));

int i;

int\* mass = (int \*)malloc(n\*sizeof(int));

if(choice==1 || choice < 1 || choice > 3){

if(choice < 1 || choice > 2) printf("\n\x1b[31mERROR: \x1b[3;31m unknown way, the array will be filled from the random\x1b[0m");

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

mass[i] = (rand()%1000)-500;

}

}

else if(choice==2){

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

printf("\nEnter element(integer number) \x1b[4;33m№%d\x1b[0m: ",i);

scanf("%d", &mass[i]);

}

}

return mass;

}

struct ret\* create\_file(){

struct ret \*rezult = (struct ret\*)malloc(sizeof(struct ret));

int\* mass;

char str[50];

int n, i,s;

printf("\n\x1b[33mATTENTION: \x1b[3;33menter the correct file name, along with the extension .txt\x1b[0m");

printf("\nenter filename : ");

scanf("%s", str);

fflush(stdin);

FILE \*myfile = fopen(str, "r");

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

mass = (int \*)malloc(n\*sizeof(int));

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

fscanf(myfile,"%d", &s);

mass[i]=s;

}

fclose(myfile);

rezult->new\_mass = mass;

rezult->per=n;

rezult->razb=0;

return rezult;

}

void print\_mass(int\* mass, int n){

printf("\nmass=[");

for(int i=0;i<n;i++){

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

}

printf("]");

}

int\* mas(int n, int\* mass){

int i;

int\* mass\_2;

mass\_2 = (int \*)malloc(n\*sizeof(int));

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

mass\_2[i]=mass[i];

}

return mass\_2;

}

void test(){

struct ret \*rez;

int\* mass;

int i,n, min=99999,max=-99999, elem;

for(n=100;n<501;n=n+100){

printf("\nN= %d", n);

min=99999;

max=-99999;

mass = (int \*)malloc(n\*sizeof(int));

for(i=0;i<n;i=i){

elem = (rand()%10000)-5000;

if(elem <= min){

mass[i] = elem;

min = elem;

i++;

}

}

printf("\n-----------\nПо убыванию\n-----------");

rez=ProstSort(n,mas(n,mass));

printf("\nТрудоемкость(Простой выбор)-%d", rez->per + rez->razb);

rez=ShelSort(n,mas(n,mass));

printf("\nТрудоемкость(Шелл)-%d", rez->per + rez->razb);

rez=InsertionSort(n,mas(n,mass));

printf("\nТрудоемкость(Бинарное включение)-%d", rez->per + rez->razb);

rez=shekerSort(n,mas(n,mass));

printf("\nТрудоемкость(Шейкерная)-%d", rez->per + rez->razb);

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

mass[i]=(rand()%10000)-5000;

}

printf("\n--------\nРандомно\n--------");

rez=ProstSort(n, mas(n,mass));

printf("\nТрудоемкость(Простой выбор)-%d", rez->per + rez->razb);

rez=ShelSort(n,mas(n,mass));

printf("\nТрудоемкость(Шелл)-%d", rez->per + rez->razb);

rez=InsertionSort(n,mas(n,mass));

printf("\nТрудоемкость(Бинарное включение)-%d", rez->per + rez->razb);

rez=shekerSort(n,mas(n,mass));

printf("\nТрудоемкость(Шейкерная)-%d", rez->per + rez->razb);

for(i=0;i<n;i=i){

elem = (rand()%10000)-5000;

if(elem >= max){

mass[i] = elem;

max = elem;

i++;

}

}

printf("\n--------------\nПо возрастанию\n--------------");

rez=ProstSort(n, mas(n,mass));

printf("\nТрудоемкость(Простой выбор)-%d", rez->per + rez->razb);

rez=ShelSort(n,mas(n,mass));

printf("\nТрудоемкость(Шелл)-%d", rez->per+rez->razb);

rez=InsertionSort(n,mas(n,mass));

printf("\nТрудоемкость(Бинарное включение)-%d", rez->per+rez->razb);

rez=shekerSort(n,mas(n,mass));

printf("\nТрудоемкость(Шейкерная)-%d", rez->per+ rez->razb);

}

free(mass);

}

------------------------------------------------------------------------------------

**Файл prost.c**

#include "header.h"

struct ret\* ProstSort(int n, int\* mass){

struct ret \*rezult = (struct ret\*)malloc(sizeof(struct ret));

rezult->new\_mass=mass;

int min,j,tmp, pere = 0, razb = 0;

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

{

min = i;

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

{

razb +=1;

if (mass[min] > mass[j]){

min = j;

}

}

tmp = mass[min];

mass[min] = mass[i];

mass[i] = tmp;

pere += 1 ;

}

rezult->per=pere;

rezult->razb=razb;

return rezult;

}

------------------------------------------------------------------------------------

**Файл Shell.c**

#include "header.h"

struct ret\* ShelSort(int n, int\* mass){

struct ret \*rezult = (struct ret\*)malloc(sizeof(struct ret));

rezult->new\_mass=mass;//массив

int i, j, pos, tmp, pere = 0, sr = 0;

for (pos = n / 2; pos > 0; pos = pos / 2){

for (i = pos; i < n; i++){

tmp = mass[i];

pere++;

for (j = i; j >= pos; j -= pos){

if (tmp < mass[j - pos]){

mass[j] = mass[j - pos];

sr++;

}

else {

break;

}

}

mass[j] = tmp;

}

}

rezult->per=pere;

rezult->razb=sr;

return rezult;

}

------------------------------------------------------------------------------------

**Файл bin.c**

#include "header.h"

struct ret\* InsertionSort(int n, int\* mass)

{

struct ret \*rezult = (struct ret\*)malloc(sizeof(struct ret));

rezult->new\_mass=mass;

int nE, pos, pere = 0, sr = 0;

for (int i = 1; i < n; i++)

{

nE = mass[i];

pos = i - 1;

while(pos >= 0 && mass[pos] > nE)

{

mass[pos+1] = mass[pos];

pos = pos - 1;

sr++;

}

mass[pos+1] = nE;

if (mass[pos + 1] = nE)

{

pere += 1;

}

}

rezult->per=pere;

rezult->razb=sr;

return rezult;

}

------------------------------------------------------------------------------------

**Файл sheyker.c**

#include "header.h"

struct ret\* shekerSort(int count, int\* mass)

{

struct ret \*rezult = (struct ret\*)malloc(sizeof(struct ret));

rezult->new\_mass=mass;

int left = 0, right = count-1;

int flag = 1, per = 0;

while ((left < right) && flag > 0)

{

for (int i = left; i<right; i++)

{

per = per + 1;

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

{

int t = mass[i];

mass[i] = mass[i + 1];

mass[i + 1] = t;

flag = flag+1;

}

}

right--;

for (int i = right; i>left; i--)

{

if (mass[i - 1]>mass[i])

{

int t = mass[i];

mass[i] = mass[i - 1];

mass[i - 1] = t;

flag = flag+1;

}

}

left++;

}

rezult->per=(flag-1);

rezult->razb=per;

return rezult;

}