Отчет по практикуму
по курсу “Фундаментальная
информатика”
Задание № 9

Студент группы М8О-111Б-23 Тимофеева Ирина Александровна,
№ по списку 21
Работа выполнена: «25» мая 2024 г.
Преподаватель: доцент каф. 806 Никулин Сергей Петрович
Отчет сдан «25» мая 2024 г., итоговая оценка _____
Подпись преподавателя
_____

1. Тема: Сортировка и поиск.

2. Цель работы: Составить программу на Си с ипользованием процедур и функций для сортировки таблицы заданным методом и двоичного поиска по ключу в таблице.

3. Задание (6/9). Сортировка (6) - Метод двоичной вставки.
Структура таблицы (9) - тип ключа: комплексный; длина ключа в байтах: 16; хранение данных и ключей: вместе; минимальное число элементов таблицы: 19

4. Оборудование *ПЭВМ студента* (*лабораторное*): ЭВМ Процессор Intel Core I7-13700H, имя узла сети User с ОП 16 ГБ, НМД 512 ГБ. Монитор: 1920x1080~60Hz
Другие устройства не использовались

5. Программное обеспечение *ПВЭМ студента* (*лабораторное*):
Операционная система семейства UNIX, наименование: Ubuntu версия 22.04.1 LTS
Интерпретатор команд: bash версия 5.1.16(1)
Система программирования: C
Редактор текстов: Emacs версия 28.1
Утилиты операционной системы: gcc, emacs
Прикладные системы и программы: emacs
Местонахождения и имена файлов программ и данных: /home/vsevolod6. Идея, метод, алгоритм решения задачи (в формах: словесной, псевдокода, графической [блоксхема, диаграмма, рисунок, таблица] или формальные спецификации с пред- и постусловиями)
Реализация заданного варианта на языке программирования Си.

7. Сценарий выполнения работы [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты либо соображения по тестированию].
Текст программы:
table.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>
#include <complex.h>
#include <math.h>
#include "table.h"

//создание таблицы
```

```
void make_table(FILE *file, Row *row)
{
        int i;
   for(i = 0; i < N; ++i){
        int re, im;
        char inf[MAXD];

      fscanf(file, "%d+%di %[^\n]\n", &re, &im, inf);
      row[i].key.real = re;
      row[i].key.image = im;
      strncpy(row[i].data, inf, MAXD);
   }
}

//печать таблицы
void Print_table(Row *row)
{
   printf("-------------------------------------------------------------------------\n");
   printf("| № | Ключ    | Данные                                                |\n");
   printf("-------------------------------------------------------------------------\n");
   int i;
   for (i = 0; i < N; i++)
        {
      printf("|%3d|%4d+%3di|%60s|\n", i, row[i].key.real, row[i].key.image, row[i].data);
        printf("-------------------------------------------------------------------------\n");
   }
}

//перемена элементов местами
void swap(Row *a, Row *b)
{
   Row temp = *a;
   *a = *b;
   *b = temp;
}

//разворот таблицы в обратном порядке
void reverse_table(Row *row)
{
        int i;
   for (i = 0; i < N / 2; i++)
        {
      swap(&row[i], &row[N-i-1]);
   }
}
```

```
//проверка существования таблицы
bool check_table(Row *row)
{
   if (row->key.real != 0 && row->key.image != 0 && row->data != "")
   {
      return true;
   }
   return false;
}

// сравнение комплексных чисел
bool comparec(Complex c1, Complex c2)
{
        //double complex f1 = c1.real + c1.image*I;
        //double complex f2 = c2.real + c2.image*I;
        double f3 = sqrt(c1.real*c1.real + c1.image*c1.image);
        double f4 = sqrt(c2.real*c2.real + c2.image*c2.image);
        return f3>f4;
}

// проверка на совпадение
bool equalc(Complex c1, Complex c2)
{
   return (c1.real == c2.real && c1.image == c2.image);
}

void shuffle_table(Row *row)
{
   srand(time(NULL));
   int i;
   for (i = 0; i < N; i++)
   {
      int posA = rand() % N;
      int posB = rand() % N;
      swap(&row[posA], &row[posB]);
   }
}

//двоичный поиск в упорядоченной таблице
int binary_search(Row *row, Complex item, int l, int r)
{
   Row search_element;
   search_element.key = item;
   while (l <= r)
```

```
        {
      int mid = l + (r - l) / 2;
      if (search_element.key.real == row[mid].key.real && search_element.key.image ==
row[mid].key.image)
                {
         return mid+1;
         break;
      }
      if (comparec(search_element.key, row[mid].key))
         l = mid + 1;
                else
         r = mid - 1;
   }
        return l;
}

// сортировка двоичными вставками
void insertion_sort(Row *row)
{
   int i, loc, j, k;
        Complex selected;

   for (i = 1; i < N; ++i) {
      j = i - 1;
      selected = row[i].key;

      // find location where selected should be inserted
      loc = binary_search(row, selected, 0, j);

      // Move all elements after location to create space
      while (j >= loc) {
        swap(&row[j+1], &row[j]);
         j--;
      }
      row[j+1].key = selected;
   }
}

//пузырьковая сортировка
void bubble_sort(Row *row)
{
        bool answer = false;
   while (answer != true)
        {
                int i;
```

```
                answer = true;
        for (i = 0; i < N - 1; i++)
        {
          if (comparec(row[i].key, row[i+1].key))
          {
                swap(&row[i], &row[i+1]);
                answer = false;
              }
        }
          }
}
```

table.h:

```
#include <stdio.h>
#include <stdbool.h>
#define N 11
#define MAXD 2000
#define FILENAME "data"

typedef struct {
   int real;
   int image;
} Complex;

typedef struct {
   Complex key;
   char data[MAXD];
} Row;

void make_table(FILE *file, Row *row);
void Print_table(Row *row);
void swap(Row *a, Row *b);
void reverse_table(Row *row);
bool check_table(Row *row);
bool comparec(Complex c1, Complex c2);
bool equalc(Complex c1, Complex c2);
void shuffle_table(Row *row);
int binary_search(Row *row, Complex item, int l, int r);
void insertion_sort(Row *row);
void bubble_sort(Row *row);
```

process_table.c:

```
#include <locale.h>
#include "table.h"
```

```
int main()
{
   setlocale(LC_ALL, "");
   Row rows[N];
   FILE *input;

   int i;
   char c;
   int done = 0;
   Complex cnum;
   while (!done)
   {
      printf("Меню:\n");
      printf("1 - создать таблицу\n");
      printf("2 - печать таблицы\n");
      printf("3 - сортировка таблицы методом двоичной вставки\n");
      printf("4 - бинарный поиск по ключу\n");
      printf("5 - развернуть таблицу\n");
      printf("6 - перемешать таблицу\n");
      printf("0 - выход\n");
      printf("Введите команду: ");
      c = getchar();
      getchar();
      switch (c)
      {
      case '0':
         done = 1;
         break;
      case '1':
         input = fopen(FILENAME, "r");
         if (input == NULL)
         {
            puts("Не удалось открыть файл\n");
            return 1;
         }
         make_table(input, rows);
         if (check_table(rows))
            printf("Таблица успешно создана!\n");
         else
            printf("Ошибка создания таблицы!\n");
         break;
      case '2':
         printf("Таблица: \n");
         Print_table(rows);
         break;
```

```
        case '3':
            insertion_sort(rows);
            printf("Выполнена сортировка таблицы!\n");
            break;
        case '4':
           printf("Введите значение ключа для поиска: ");
                    scanf("%d+%di", &cnum.real, &cnum.image);
            i = binary_search(rows, cnum, 0, N-1);
            if (equalc(cnum, rows[i-1].key))
                    printf("Выполнен поиск - найден элемент [%d+%di] с индексом %d\n",
rows[i-1].key.real, rows[i-1].key.image, i-1);
          else
                    printf("Выполнен поиск - элемент не найден!\n");
            getchar();
            break;
        case '5':
            reverse_table(rows);
            printf("Выполнена инверсия таблицы!\n");
            break;
        case '6':
            shuffle_table(rows);
            printf("Таблица перемешана в случайном порядке!\n");
            break;
        default:
            printf("ошибка ввода!\n");
            printf("\n");
        }
    }
    return 0;
}
```

8. Распечатка протокола (подклеить листинг окончательного варианта программы с тестовыми примерами, подписанный преподавателем).

```
#листинг программы работы с таблицами
#заголовочный файл
irina@Irina-VivoBook:~/Prog/Prog_C/Lab8$ cat table.h
#include <stdio.h>
#include <stdbool.h>
#define N 11
#define MAXD 2000
#define FILENAME "data"

typedef struct {
    int real;
    int image;
```

```
} Complex;

typedef struct {
   Complex key;
   char data[MAXD];
} Row;

void make_table(FILE *file, Row *row);
void Print_table(Row *row);
void swap(Row *a, Row *b);
void reverse_table(Row *row);
bool check_table(Row *row);
bool comparec(Complex c1, Complex c2);
bool equalc(Complex c1, Complex c2);
void shuffle_table(Row *row);
int binary_search(Row *row, Complex item, int l, int r);
void insertion_sort(Row *row);
void bubble_sort(Row *row);

#библиотека функций
irina@Irina-VivoBook:~/Prog/Prog_C/Lab8$ cat table.c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>
#include <complex.h>
#include <math.h>
#include "table.h"

//создание таблицы
void make_table(FILE *file, Row *row)
{
        int i;
   for(i = 0; i < N; ++i){
        int re, im;
        char inf[MAXD];

      fscanf(file, "%d+%di %[^\n]\n", &re, &im, inf);
      row[i].key.real = re;
      row[i].key.image = im;
      strncpy(row[i].data, inf, MAXD);
   }
}

//печать таблицы
```

```
void Print_table(Row *row)
{
   printf("-------------------------------------------------------------------------\n");
   printf("| № | Ключ    | Данные                                                |\n");
   printf("-------------------------------------------------------------------------\n");
   int i;
   for (i = 0; i < N; i++)
         {
      printf("|%3d|%4d+%3di|%60s|\n", i, row[i].key.real, row[i].key.image, row[i].data);
        printf("-------------------------------------------------------------------------\n");
   }
}

//перемена элементов местами
void swap(Row *a, Row *b)
{
   Row temp = *a;
   *a = *b;
   *b = temp;
}

//разворот таблицы в обратном порядке
void reverse_table(Row *row)
{
         int i;
   for (i = 0; i < N / 2; i++)
         {
      swap(&row[i], &row[N-i-1]);
   }
}

//проверка существования таблицы
bool check_table(Row *row)
{
   if (row->key.real != 0 && row->key.image != 0 && row->data != "")
   {
      return true;
   }
   return false;
}

// сравнение комплексных чисел
bool comparec(Complex c1, Complex c2)
{
         //double complex f1 = c1.real + c1.image*I;
```

```
        //double complex f2 = c2.real + c2.image*I;
        double f3 = sqrt(c1.real*c1.real + c1.image*c1.image);
        double f4 = sqrt(c2.real*c2.real + c2.image*c2.image);
        return f3>f4;
}

// проверка на совпадение
bool equalc(Complex c1, Complex c2)
{
   return (c1.real == c2.real && c1.image == c2.image);
}

void shuffle_table(Row *row)
{
   srand(time(NULL));
   int i;
   for (i = 0; i < N; i++)
   {
      int posA = rand() % N;
      int posB = rand() % N;
      swap(&row[posA], &row[posB]);
   }
}

//двоичный поиск в упорядоченной таблице
int binary_search(Row *row, Complex item, int l, int r)
{
   Row search_element;
   search_element.key = item;
   while (l <= r)
         {
      int mid = l + (r - l) / 2;
      if (search_element.key.real == row[mid].key.real && search_element.key.image ==
row[mid].key.image)
                  {
         return mid+1;
         break;
      }
      if (comparec(search_element.key, row[mid].key))
         l = mid + 1;
                  else
         r = mid - 1;
   }
        return l;
}
```

```
// сортировка двоичными вставками
void insertion_sort(Row *row)
{
   int i, loc, j, k;
        Complex selected;

   for (i = 1; i < N; ++i) {
      j = i - 1;
      selected = row[i].key;

      // find location where selected should be inserted
      loc = binary_search(row, selected, 0, j);

      // Move all elements after location to create space
      while (j >= loc) {
        swap(&row[j+1], &row[j]);
         j--;
      }
      row[j+1].key = selected;
   }
}

//пузырьковая сортировка
void bubble_sort(Row *row)
{
        bool answer = false;
   while (answer != true)
        {
                int i;
                answer = true;
      for (i = 0; i < N - 1; i++)
      {
        if (comparec(row[i].key, row[i+1].key))
        {
                swap(&row[i], &row[i+1]);
                answer = false;
              }
      }
        }
}

# основная программа
irina@Irina-VivoBook:~/Prog/Prog_C/Lab8$ cat process_table.c
#include <locale.h>
```

```
#include "table.h"

int main()
{
   setlocale(LC_ALL, "");
   Row rows[N];
   FILE *input;

   int i;
   char c;
   int done = 0;
   Complex cnum;
   while (!done)
   {
      printf("Меню:\n");
      printf("1 - создать таблицу\n");
      printf("2 - печать таблицы\n");
      printf("3 - сортировка таблицы методом двоичной вставки\n");
      printf("4 - бинарный поиск по ключу\n");
      printf("5 - развернуть таблицу\n");
      printf("6 - перемешать таблицу\n");
      printf("0 - выход\n");
      printf("Введите команду: ");
      c = getchar();
      getchar();
      switch (c)
      {
      case '0':
         done = 1;
         break;
      case '1':
         input = fopen(FILENAME, "r");
         if (input == NULL)
         {
            puts("Не удалось открыть файл\n");
            return 1;
         }
         make_table(input, rows);
         if (check_table(rows))
            printf("Таблица успешно создана!\n");
         else
            printf("Ошибка создания таблицы!\n");
         break;
      case '2':
         printf("Таблица: \n");
```

```
            Print_table(rows);
            break;
        case '3':
            insertion_sort(rows);
            printf("Выполнена сортировка таблицы!\n");
            break;
        case '4':
           printf("Введите значение ключа для поиска: ");
                    scanf("%d+%di", &cnum.real, &cnum.image);
            i = binary_search(rows, cnum, 0, N-1);
            if (equalc(cnum, rows[i-1].key))
                    printf("Выполнен поиск - найден элемент [%d+%di] с индексом %d\n",
rows[i-1].key.real, rows[i-1].key.image, i-1);
           else
                    printf("Выполнен поиск - элемент не найден!\n");
            getchar();
            break;
        case '5':
            reverse_table(rows);
            printf("Выполнена инверсия таблицы!\n");
            break;
        case '6':
            shuffle_table(rows);
            printf("Таблица перемешана в случайном порядке!\n");
            break;
        default:
            printf("ошибка ввода!\n");
            printf("\n");
        }
    }
    return 0;
}

#компиляция программы
irina@Irina-VivoBook:~/Prog/Prog_C/Lab9$ gcc *.c -lm -o proc_table
table.c: In function ‘make_table’:
table.c:20:9: warning: implicit declaration of function ‘strncpy’ [-Wimplicit-function-
declaration]
   20 |         strncpy(row[i].data, inf, MAXD);
      |         ^~~~~~
table.c:8:1: note: include ‘<string.h>’ or provide a declaration of ‘strncpy’
    7 | #include "table.h"
  +++ |+#include <string.h>
    8 |
```

```
table.c:20:9: warning: incompatible implicit declaration of built-in function ‘strncpy’ [-
Wbuiltin-declaration-mismatch]
   20 |         strncpy(row[i].data, inf, MAXD);
      |         ^~~~~~~
table.c:20:9: note: include ‘<string.h>’ or provide a declaration of ‘strncpy’

irina@Irina-VivoBook:~/Prog/Prog_C/Lab9$ ls
data  process_table.c  proc_table  table.c  table.h

#запуск программы
irina@Irina-VivoBook:~/Prog/Prog_C/Lab9$ ./proc_table
Меню:
1 - создать таблицу
2 - печать таблицы
3 - сортировка таблицы методом двоичной вставки
4 - бинарный поиск по ключу
5 - развернуть таблицу
6 - перемешать таблицу
0 - выход
Введите команду: 1
Таблица успешно создана!
Меню:
1 - создать таблицу
2 - печать таблицы
3 - сортировка таблицы методом двоичной вставки
4 - бинарный поиск по ключу
5 - развернуть таблицу
6 - перемешать таблицу
0 - выход
Введите команду: 2
Таблица:
-------------------------------------------------------------------------
| № | Ключ    | Данные                                                    |
-------------------------------------------------------------------------
|  0|   5+  6i|                                                   gamma
-------------------------------------------------------------------------
|  1|  15+ 16i|                                                    teta
-------------------------------------------------------------------------
|  2|   7+  8i|                                                   delta
-------------------------------------------------------------------------
|  3|   3+  4i|                                                    beta
-------------------------------------------------------------------------
|  4|  11+ 12i|                                                    zeta
-------------------------------------------------------------------------
|  5|   1+  2i|                                                   alpha
```

```
-------------------------------------------------------------------------
|  6|  19+ 20i|                                                  kappa
-------------------------------------------------------------------------
|  7|  21+ 22i|                                                 lambda
-------------------------------------------------------------------------
|  8|  13+ 14i|                                                    eta
-------------------------------------------------------------------------
|  9|   9+ 10i|                                                epsilon
-------------------------------------------------------------------------
| 10|  17+ 18i|                                                   yota
-------------------------------------------------------------------------
Меню:
1 - создать таблицу
2 - печать таблицы
3 - сортировка таблицы методом двоичной вставки
4 - бинарный поиск по ключу
5 - развернуть таблицу
6 - перемешать таблицу
0 - выход
Введите команду: 3
Выполнена сортировка таблицы!
Меню:
1 - создать таблицу
2 - печать таблицы
3 - сортировка таблицы методом двоичной вставки
4 - бинарный поиск по ключу
5 - развернуть таблицу
6 - перемешать таблицу
0 - выход
Введите команду: 2
Таблица:
-------------------------------------------------------------------------
| № | Ключ    | Данные                                                |
-------------------------------------------------------------------------
|  0|   1+  2i|                                                  alpha
-------------------------------------------------------------------------
|  1|   3+  4i|                                                   beta
-------------------------------------------------------------------------
|  2|   5+  6i|                                                  gamma
-------------------------------------------------------------------------
|  3|   7+  8i|                                                  delta
-------------------------------------------------------------------------
|  4|   9+ 10i|                                                epsilon
-------------------------------------------------------------------------
|  5|  11+ 12i|                                                   zeta
```

```
--------------------------------------------------------------------------
|  6|  13+ 14i|                                                      eta
--------------------------------------------------------------------------
|  7|  15+ 16i|                                                     teta
--------------------------------------------------------------------------
|  8|  17+ 18i|                                                     yota
--------------------------------------------------------------------------
|  9|  19+ 20i|                                                    kappa
--------------------------------------------------------------------------
| 10|  21+ 22i|                                                   lambda
--------------------------------------------------------------------------
Меню:
1 - создать таблицу
2 - печать таблицы
3 - сортировка таблицы методом двоичной вставки
4 - бинарный поиск по ключу
5 - развернуть таблицу
6 - перемешать таблицу
0 - выход
Введите команду: 4
Введите значение ключа для поиска: 7+8i
Выполнен поиск - найден элемент [7+8i] с индексом 3
Меню:
1 - создать таблицу
2 - печать таблицы
3 - сортировка таблицы методом двоичной вставки
4 - бинарный поиск по ключу
5 - развернуть таблицу
6 - перемешать таблицу
0 - выход
Введите команду: 4
Введите значение ключа для поиска: 2+9i
Выполнен поиск - элемент не найден!
Меню:
1 - создать таблицу
2 - печать таблицы
3 - сортировка таблицы методом двоичной вставки
4 - бинарный поиск по ключу
5 - развернуть таблицу
6 - перемешать таблицу
0 - выход
Введите команду: 5
Выполнена инверсия таблицы!
Меню:
1 - создать таблицу
```

2 - печать таблицы
3 - сортировка таблицы методом двоичной вставки
4 - бинарный поиск по ключу
5 - развернуть таблицу
6 - перемешать таблицу
0 - выход
Введите команду: 2
Таблица:

№	Ключ	Данные
0	21+ 22i	lambda
1	19+ 20i	kappa
2	17+ 18i	yota
3	15+ 16i	teta
4	13+ 14i	eta
5	11+ 12i	zeta
6	9+ 10i	epsilon
7	7+ 8i	delta
8	5+ 6i	gamma
9	3+ 4i	beta
10	1+ 2i	alpha

Меню:
1 - создать таблицу
2 - печать таблицы
3 - сортировка таблицы методом двоичной вставки
4 - бинарный поиск по ключу
5 - развернуть таблицу
6 - перемешать таблицу
0 - выход
Введите команду: 6
Таблица перемешана в случайном порядке!
Меню:
1 - создать таблицу

2 - печать таблицы
3 - сортировка таблицы методом двоичной вставки
4 - бинарный поиск по ключу
5 - развернуть таблицу
6 - перемешать таблицу
0 - выход
Введите команду: 2
Таблица:

№	Ключ	Данные
0	11+ 12i	zeta
1	19+ 20i	kappa
2	21+ 22i	lambda
3	3+ 4i	beta
4	13+ 14i	eta
5	17+ 18i	yota
6	9+ 10i	epsilon
7	7+ 8i	delta
8	1+ 2i	alpha
9	15+ 16i	teta
10	5+ 6i	gamma

Меню:
1 - создать таблицу
2 - печать таблицы
3 - сортировка таблицы методом двоичной вставки
4 - бинарный поиск по ключу
5 - развернуть таблицу
6 - перемешать таблицу
0 - выход
Введите команду: 0

9. Дневник отладки должен содержать дату и время сеансов отладки и основные события (ошибки в сценарии и программе, нестандартные ситуации) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других

ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

10. Замечания автора по существу работы

11. Выводы:
Научилась реализовывать сортировки, работать с таблицами по ключу.

Подпись студента _____