Московский авиационный институт
(Национальный исследовательский университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Задание IX по курсовому проекту

«Сортировка и поиск»

Руководитель: Никулин С.П.

_____
(оценка)

_____ _____
(подпись) (дата)

Студент группы 8О-104Б: Дюсекеев Алишер

Москва 2018

Постановка задачи:

Составить программу на языке Си с использованием процедур и функций для сортировки таблицы заданным методом и двоичного поиска по ключу в таблице.

Программа должна вводить значения элементов неупорядоченной таблицы и проверять работу процедуры сортировки в трех случаях:

1. элементы таблицы с самого начала упорядочены;

2. элементы таблицы расставлены в обратном порядке;

3. элементы таблицы не упорядочены.

В последнем случае можно использовать встроенные процедуры генерации псевдослучайных чисел.

Для каждого вызова процедуры сортировки необходимо печатать исходное состояние таблицы и результаты сортировки. После выполнения сортировки программа должна вводить ключи и для каждого из них выполнять поиск в

упорядоченной таблице с помощью процедуры двоичного поиска и печатать найденные элементы, если они присутствуют в таблице.

В качестве текста для записей таблицы взять фрагмент стихотворения. Каждый элемент таблицы, содержащий ключ и текст записи, распечатывать в отдельной строке.

Метод сортировки:

9. Пирамидальная сортировка с просеиванием.

Структура таблицы:

№	Тип ключа	Длина ключа в байтах	Хранение данных и ключей	Число элементов таблицы
9	Комплексный	16	Вместе	8-10

Общий метод решения:

Таблица

Так как ключ комплексный, то он является составным, поэтому удобнее всего представлять его в виде структуры, состоящей из двух переменных типа double, где Re – действительная, а Im – мнимая часть ключа.

```
typedef struct
{
	double Re;
	double Im;
}Key;
```

Так как хранение данных и ключей вместе, причем данные хранятся в массиве типа char, то представлять ячейку таблицы будем в виде структуры:

```
typedef struct
{
	Key key;
	char s[60];
}Cell;
```

Таким образом, зная размер таблицы по условию, можно представить ее в виде массива ячеек Cell:

```
Cell T[11];
```

Пирамидальная сортировка

Сортировка пирамидой использует бинарное сортирующее дерево. Сортирующее дерево — это такое дерево, у которого выполнены условия:

1. Каждый лист имеет глубину либо d, либо d-1, d — максимальная глубина дерева.
2. Значение в любой вершине не меньше (другой вариант — не больше) значения её потомков.

Удобная структура данных для сортирующего дерева — такой массив Array, что Array[0] — элемент в корне, а потомки элемента Array[i] являются Array[2i+1] и Array[2i+2].

Алгоритм сортировки будет состоять из двух основных шагов:

1. Выстраиваем элементы массива в виде сортирующего дерева

Array[i] >= Array[2i+1]

Array[i] >= Array[2i+2]

при 0 <= i <= n/2

2. Будем удалять элементы из корня по одному за раз и перестраивать дерево. То есть на первом шаге обмениваем Array[0] и Array[n-1], преобразовываем Array[0], Array[1], …, Array[n-2] в сортирующее дерево. Затем переставляем Array[0] и Array[n-2], преобразовываем Array[0], Array[1], … , Array[n-3] в сортирующее дерево. Процесс продолжается до тех пор, пока в сортирующем дереве не останется один элемент. Тогда Array[0], Array[1], … , Array[n-1] — упорядоченная последовательность. Этот шаг требует O(n*log(n)) операций.

Сортировка ячеек таблицы осуществляется по модулю комплексного ключа.

Бинарный поиск

1. Определение значения элемента в середине структуры данных. Полученное значение сравнивается с ключом.
2. Если ключ меньше значения середины, то поиск осуществляется в первой половине элементов, иначе — во второй.

3. Поиск сводится к тому, что вновь определяется значение серединного элемента в выбранной половине и сравнивается с ключом.
4. Процесс продолжается до тех пор, пока не будет найден элемент со значением ключа или не станет пустым интервал для поиска.

Общие сведения о программе:

Программное и аппаратное обеспечение для запуска данной программы на ПК не ограничено в выборе. Операционная система семейства Windows NT – Windows 10. Интерпретатор команд – cmd.exe. Интегрированная среда разработки – Visual Studio Express 2013. Язык программирования – C. Строк в программе – 191. Местонахождение текстового файла: C:\Users\Алишер\Desktop\прога\kp9. Имя файлов: kp9.c. Программа запускается через Visual Studio или вручную, после компиляции через командную строку через приложения kp9.exe.

Описание логической структуры:

1. Создаем текстовый файл, содержащий таблицу.
2. Запускаем приложение kp9.exe.
3. С таблицей можно произвести 5 действий:
   3.1. Печать таблицы.
   3.2. Обратный порядок.
   3.3. Перемешать.
   3.4. Поиск по ключу.
   3.5. Сортировка.

Таблица переменных и констант:

Название переменной	**Тип переменной**	**Назначение переменной**
Re	double	Действительная часть ключа.
Im	double	Мнимая часть ключа.
key	struct Key	Ключ.
s	char[]	Данные.

T	struct Cell[]	Таблица.
f	FILE*	Текстовый файл.
what	struct key	Ключ, по которому осуществляется поиск.
size	int	Размер таблицы.
o	int	Индекс, ключа what.
tmp	struct Cell	Временная ячейка, в которую копируются элементы таблицы.
k	int	Переменная выбора в меню.
i	int	Счетчик.

Дополнительные функции:

- void readfile(FILE* f, Cell *T, int* size)
  Обрабатывает текстовый файл и записывает его в массив.
- void PrintTable(Cell* T, int size)
  Печать таблицы.
- void Reverse(Cell *T, int size)
  Перемешивает таблицу в обратном порядке.
- void Random(Cell *T, int size)
  Перемешивает таблицу в случайном порядке.
- double Module(Key V)
  Возвращает модуль комплексного ключа.
- int BinSearch(Cell* T, int size, Key what)
  Бинарный поиск. Возвращает индекс найденной ячейки, в противном случае -1.
- int Checksort(Cell* T, int size)
  Проверка таблицы на отсортированность.
- void Sort(Cell* T, int size)
  Сортировка таблицы.
- void Downheap(Cell* T,int k, int n)
  Просеивает элемент сквозь пирамиду при сортировке.
- int menu()
  Функция, которая выводит меню на экран и проверяет корректность выбора опции.

Входные данные:

Входные представляются в виде текстового файла input.txt, где первые два столбца являются значениями ключа, а последний соответственно данными в виде строк стихотворения.

input.txt

1 1 Я всматриваюсь в вас, о, числа,

2 1 И вы мне видитесь одетыми в звери, в их шкурах,

2 -2 Рукой опирающимися на вырванные дубы.

3 2 Вы даруете единство между змееобразным движением

3 3 Хребта вселенной и пляской коромысла,

4 -3 Вы позволяете понимать века, как быстрого хохота зубы.

4 -4 Мои сейчас вещеобразно разверзлися зеницы

5 4 Узнать, что будет Я, когда делимое его - единица.

Далее входные данные вводятся с экрана в виде вещественных чисел, если выбрана опция 4 поиска по ключу в меню.

Выходные данные:

Выходные данные зависят от выбора опции в меню, если выбрано 1, то таблица выводится на экран, если 4, то только ее ячейка, если ключ найден.

Программа на Си:

kp9.c

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include<math.h>
#include<stdlib.h>
#include<string.h>
#include<Windows.h>
#include<time.h>

typedef struct
{
	double Re;
	double Im;
}Key;
typedef struct
{
	Key key;
	char s[60];
}Cell;
void readfile(FILE* f, Cell *T, int* size)//+
{
	*size = 0;
```

```
	while (!feof(f))
	{
		fscanf(f, "%lf\t%lf\t", &T[*size].key.Re, &T[*size].key.Im);
		fgets(T[*size].s, 60, f);
		(*size)++;
	}
	(*size)--;
}
void PrintTable(Cell* T, int size)//+
{
	int i;
	printf("-----------------------------------------------------------------\n"
		"|Key |Text\n-----------------------------------------------------------------
----\n");
	for (i = 0; i < size; i++)
	{
		printf("|%g%+gi|%s", T[i].key.Re, T[i].key.Im, T[i].s);
	}
	printf("-----------------------------------------------------------------\n");
}
void Reverse(Cell *T, int size)//+
{
	int i;
	Cell tmp;
	for (i = 0; i < size/2; i++)
	{
		tmp = T[i];
		T[i] = T[size - 1 - i];
		T[size - 1 - i] = tmp;
	}
}
void Random(Cell *T, int size)//+
{
	srand(time(NULL));
	Cell tmp;
	int i;
	int r = rand() % size;
	for (i = 0; i < size; i++, r = (r + i * 3) % size)
	{
		tmp = T[i];
		T[i] = T[r];
		T[r] = tmp;
	}
}
double Module(Key V)
{
	return sqrt(V.Im * V.Im + V.Re * V.Re);
}
int BinSearch(Cell* T, int size, Key what)//+
{
	int m, diff, first = 0, last = size;
	for (;;)
	{
		diff = last - first;
		m = first + diff / 2;
		if (diff < 0)
			return -1;
		if (T[m].key.Im == what.Im && T[m].key.Re == what.Re)
			return m;
		if (Module(T[m].key)<Module(what))
			first = m + 1;
		else
			last = m - 1;
	}
}
```

```
void Downheap(Cell* T,int k, int n)
{
	Cell tmp;
	int child;
	tmp = T[k];
	while (k <= n / 2)
	{
		child = 2 * k;
		if (child < n && Module(T[child].key) < Module(T[child+1].key))
			child++;
		if (Module(tmp.key) >= Module(T[child].key))
			break;
		T[k] = T[child];
		k = child;
	}
	T[k] = tmp;
}
void Sort(Cell* T, int size)
{
	int i;
	Cell tmp;
	for (i = size / 2 - 1; i >= 0; i--)
		Downheap(T, i, size - 1);
	for (i = size - 1; i > 0; i--)
	{
		tmp = T[i];
		T[i] = T[0];
		T[0] = tmp;
		Downheap(T, 0, i - 1);
	}
}
int Checksort(Cell* T, int size)
{
	int i;
	for (i = 0; i < size - 1; i++)
	{
		if (Module(T[i].key)>Module(T[i + 1].key))
			return 1;
	}
	return 0;
}
int menu()
{
	int k;
	printf("0:Закончить работу\n");
	printf("1:Печать таблицы\n");
	printf("2:Обратный порядок\n");
	printf("3:Перемешать\n");
	printf("4:Поиск по ключу\n");
	printf("5:Сортировка\n");
	printf(":>");
	scanf("%d", &k);
	while (k < 0 || k>5)
	{
		printf("Ошибка. Вы ввели номер действия, которого не существует\n"
			"Попробуйте ввести снова\n:>");
		scanf("%d", &k);
	}
	return k;
}
int main()
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	int size;
```

```
	Cell T[11];
	FILE* f = fopen("input.txt", "r");
	if (!f)
	{
		perror("Can't open file");
		return 2;
	}
	readfile(f, T, &size);
	while (1)
	{
		switch (menu())
		{
		case 0:return 0;
		case 1:
			PrintTable(T, size); break;
		case 2:
			Reverse(T, size); break;
		case 3:
			Random(T, size); break;
		case 4:
			if (Checksort(T, size) == 1)
				puts("Таблица неотсортирована\n");
			else
			{
				int o;
				Key what;
				puts("Введите ключ:\n");
				scanf("%lf%lf", &what.Re, &what.Im);
				o = BinSearch(T, size, what);
				if (o == -1)
					puts("Ключ не найден\n");
				else
					printf("|%g%+gi|%s", T[o].key.Re, T[o].key.Im, T[o].s);
			}
			break;
		case 5:
			Sort(T, size); break;
		}
	}
	return 0;
}
```

Демонстрация работы программы:

```
0:Закончить работу

1:Печать таблицы

2:Обратный порядок

3:Перемешать

4:Поиск по ключу

5:Сортировка

:>1
```

```
---------------------------------------------------------------
|Key |Text
---------------------------------------------------------------
|1+1i|Я всматриваюсь в вас, о, числа,
|2+1i|И вы мне видитесь одетыми в звери, в их шкурах,
|2-2i|Рукой опирающимися на вырванные дубы.
|3+2i|Вы даруете единство между змееобразным движением
|3+3i|Хребта вселенной и пляской коромысла,
|4-3i|Вы позволяете понимать века, как быстрого хохота зубы.
|4-4i|Мои сейчас вещеобразно разверзлися зеницы
|5+4i|Узнать, что будет Я, когда делимое его - единица.
---------------------------------------------------------------
0:Закончить работу
1:Печать таблицы
2:Обратный порядок
3:Перемешать
4:Поиск по ключу
5:Сортировка
:>2
0:Закончить работу
1:Печать таблицы
2:Обратный порядок
3:Перемешать
4:Поиск по ключу
5:Сортировка
:>1
---------------------------------------------------------------
|Key |Text
---------------------------------------------------------------
|5+4i|Узнать, что будет Я, когда делимое его - единица.
|4-4i|Мои сейчас вещеобразно разверзлися зеницы
|4-3i|Вы позволяете понимать века, как быстрого хохота зубы.
```

```
|3+3i|Хребта вселенной и пляской коромысла,
|3+2i|Вы даруете единство между змееобразным движением
|2-2i|Рукой опирающимися на вырванные дубы.
|2+1i|И вы мне видитесь одетыми в звери, в их шкурах,
|1+1i|Я всматриваюсь в вас, о, числа,
----------------------------------------------------------------
0:Закончить работу
1:Печать таблицы
2:Обратный порядок
3:Перемешать
4:Поиск по ключу
5:Сортировка
:>3
0:Закончить работу
1:Печать таблицы
2:Обратный порядок
3:Перемешать
4:Поиск по ключу
5:Сортировка
:>1
----------------------------------------------------------------
|Key |Text
----------------------------------------------------------------
|4-4i|Мои сейчас вещеобразно разверзлися зеницы
|3+3i|Хребта вселенной и пляской коромысла,
|5+4i|Узнать, что будет Я, когда делимое его - единица.
|3+2i|Вы даруете единство между змееобразным движением
|4-3i|Вы позволяете понимать века, как быстрого хохота зубы.
|2+1i|И вы мне видитесь одетыми в звери, в их шкурах,
|1+1i|Я всматриваюсь в вас, о, числа,
|2-2i|Рукой опирающимися на вырванные дубы.
----------------------------------------------------------------
```

```
0:Закончить работу
1:Печать таблицы
2:Обратный порядок
3:Перемешать
4:Поиск по ключу
5:Сортировка
:>4
Таблица неотсортирована
0:Закончить работу
1:Печать таблицы
2:Обратный порядок
3:Перемешать
4:Поиск по ключу
5:Сортировка
:>5
0:Закончить работу
1:Печать таблицы
2:Обратный порядок
3:Перемешать
4:Поиск по ключу
5:Сортировка
:>1
------------------------------------------------------------------
|Key |Text
------------------------------------------------------------------
|1+1i|Я всматриваюсь в вас, о, числа,
|2+1i|И вы мне видитесь одетыми в звери, в их шкурах,
|2-2i|Рукой опирающимися на вырванные дубы.
|3+2i|Вы даруете единство между змееобразным движением
|3+3i|Хребта вселенной и пляской коромысла,
|4-3i|Вы позволяете понимать века, как быстрого хохота зубы.
|4-4i|Мои сейчас вещеобразно разверзлися зеницы
```

```
|5+4i|Узнать, что будет Я, когда делимое его - единица.
----------------------------------------------------------------------
0:Закончить работу
1:Печать таблицы
2:Обратный порядок
3:Перемешать
4:Поиск по ключу
5:Сортировка
:>4
Введите ключ:
2 -2
|2-2i|Рукой опирающимися на вырванные дубы.
0:Закончить работу
1:Печать таблицы
2:Обратный порядок
3:Перемешать
4:Поиск по ключу
5:Сортировка
:>4
Введите ключ:
2 1
|2+1i|И вы мне видитесь одетыми в звери, в их шкурах,
0:Закончить работу
1:Печать таблицы
2:Обратный порядок
3:Перемешать
4:Поиск по ключу
5:Сортировка
:>4
Введите ключ:
9 3
Ключ не найден
```

```
0:Закончить работу

1:Печать таблицы

2:Обратный порядок

3:Перемешать

4:Поиск по ключу

5:Сортировка

:>0

Для продолжения нажмите любую клавишу . . .
```

Вывод:

Двоичный поиск - эффективный метод. Если, например, длина массива равна 1023, после первого сравнения область сужается до 511 элементов, а после второй - до 255. Легко посчитать, что для поиска в массиве из 1023 элементов достаточно 10 сравнений.

Несмотря на некоторую внешнюю сложность, пирамидальная сортировка является одной из самых эффективных. Алгоритм сортировки эффективен для больших **n**. В худшем случае требуется $\mathbf{n \cdot \log_2 n}$ шагов, сдвигающих элементы. Среднее число перемещений примерно равно $\mathbf{(n/2) \cdot \log_2 n}$, и отклонения от этого значения относительно невелики.