Основы алгоритмизации и программирования

Лабораторная работа №6.

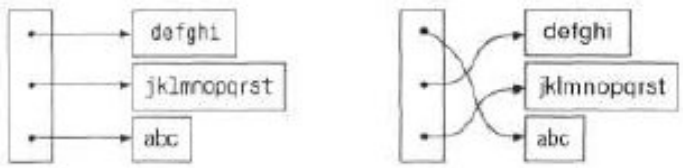
Цель работы - изучить многомерные массивы в языке программирования Си.
Содержание отчёта:

1. Титульный лист
2. Формулировка цели и задач работы
3. Описание результатов выполнения
4. Выводы, согласованные с целью

Массивы указателей

Как и любые другие переменные, указатели можно группировать в массивы. Для иллюстрации этого напишем программу, сортирующую в алфавитном порядке текстовые строки; это будет упрощенный вариант программы sort системы UNIX.

Поскольку строки в памяти расположены вплотную друг к другу, к каждой отдельной строке доступ просто осуществлять через указатель на ее первый символ. Сами указатели можно организовать в виде массива. Одна из возможностей сравнить две строки — передать указатели на них функции strcmp. Чтобы поменять местами строки, достаточно будет поменять местами в массиве их указатели (а не сами строки).



Здесь снимаются сразу две проблемы: одна — связанная со сложностью управления памятью, а вторая - с большими накладными расходами при перестановках самих строк.

Процесс сортировки распадается на три этапа:

1. чтение всех строк из ввода
2. сортировка введенных строк
3. печать их по порядку

Как обычно, выделим функции, соответствующие естественному делению задачи, и напишем главную программу main, управляющую этими функциями. Отложим на время реализацию этапа сортировки и сосредоточимся на структуре данных и вводе-выводе.

Программа ввода должна прочитать и запомнить символы всех строк, а также построить массив указателей на строки. Она, кроме того, должна подсчитать число введенных строк — эта информация понадобится для сортировки и печати. Так как функция ввода может, работать только с конечным числом строк, то, если их введено слишком много, она будет выдавать некоторое значение, которое никогда не совпадет с количеством строк, например -1.

Программа вывода занимается только тем, что печатает строки, причем в том порядке, в котором расположены указатели на них в массиве.

```
#include <stdio.h>
#include <string.h>
```

```
#include <stdlib.h>

#define MAXLINES 5000 /* максимальное число строк */

char *lineptr[MAXLINES]; /* указатели на строки */

int readlines(char *lineptr[], int nlines);
void writelines(char *lineptr[], int nlines);

void qsort(char *lineptr[], int left, int right); /* сортировка строк */

int main() {
	int i, nlines; /* количество прочитанных строк */
	if ((nlines = readlines(lineptr, MAXLINES)) >= 0) {
		qsort(lineptr, 0, nlines-1);
		writelines(lineptr, nlines); return 0;
	} else {
		printf("ошибка: слишком много строк\n");
		return 1;
	}
	/* Освобождение памяти */
	for (i = 0; i < nlines; ++i) {
		free(lineptr[i]);
	}
}

#define MAXLEN 1000 /* максимальная длина строки */

int readlines(char *lineptr[], int maxlines) {
	int len, nlines;
	char *p, line[MAXLEN];
	nlines = 0;
	while (fgets(line, MAXLEN, stdin) != NULL)
		len = strlen(line);
		if (len <= 0) {
			break;
		}
		/* Выделение памяти */
		if (nlines >= maxlines || (p = malloc(len+1)) == NULL) {
			return -1;
		} else {
			line[len-1] = '\0'; /* убираем символ \n */
			strcpy(p, line);
			lineptr[nlines++] = p;
		}
	return nlines; /* writelines: печать строк */
}

void writelines(char *lineptr[], int nlines) {
	int i;
	for (i = 0; i < nlines; i++) {
		printf("%s\n", lineptr[i]);
```

```
        }
}
```

Основное новшество здесь — объявление lineptr: char *lineptr[MAXLINES] в котором сообщается, что lineptr есть массив из MAXLINES элементов, каждый из которых представляет собой указатель на char. Иначе говоря, lineptr[i] — указатель на символ, а *lineptr[i] — символ, на который он указывает (первый символ i-й строки текста).

Так как lineptr — имя массива, его можно трактовать как указатель, т. е. так же, как мы это делали в предыдущих примерах, и writelines переписать следующим образом:

```
/* writelines: печать строк */
void writelines(char *lineptr[], int nlines) {
        while (nlines-- > 0) {
                printf( "%s\n", *lineptr++);
        }
}
```

Вначале *lineptr указывает на первую строку; каждое приращение указателя приводит к тому, что *lineptr указывает на следующую строку, и делается это до тех пор, пока nlines не станет нулем.

Теперь, когда мы разобрались с вводом и выводом, можно приступить к сортировке.

```
/* qsort: сортирует v[left]...v[right] по возрастанию */
void qsort(char *v[], int left, int right) {
        int i, last;
        void swap(char *v[], int i, int j);
        if (left >= right) /* ничего не делается, если в массиве */
                return; /* менее двух элементов */
        swap(v, left, (left+ right)/2);
        last = left;
        for (i = left+1; i <= right; i++) {
                if (strcmp(v[i], v[left]) < 0) {
                        swap(v, ++last, i);
                }
        }
        swap(v, left, last);
        qsort(v, left, last-1);
        qsort(v, last+1, right);
}
```

Небольшие поправки требуются и в программе перестановки.

```
/* swap: поменять местами v[i] и v[j] */
void swap(char *v[], int i, int j) {
        char *temp;
        temp = v[i];
        v[i] = v[j];
        v[j] = temp;
}
```

Так как каждый элемент массива v (т. е. lineptr) является указателем на символ, temp должен иметь тот же тип, что и v — тогда можно будет осуществлять пересылки между temp и элементами v.

Многомерные массивы

В Си имеется возможность задавать прямоугольные многомерные массивы, правда, на практике по сравнению с массивами указателей они используются значительно реже. В этом параграфе мы продемонстрируем некоторые их свойства.

Рассмотрим задачу перевода даты "день-месяц" в "день года" и обратно. Например, 1 марта — это 60-й день невисокосного или 61-й день високосного года. Определим две функции для этих преобразований: функция day_of_year будет преобразовывать месяц и день в день года, а month_day — день года в месяц и день. Поскольку последняя функция вычисляет два значения, аргументы месяц и день будут указателями. Так вызов month_day(1988, 60, &m, &d) присваивает переменной m значение 2, а d — 29 (29 февраля).

Нашим функциям нужна одна и та же информация, а именно таблица, содержащая числа дней каждого месяца. Так как для високосного и невисокосного годов эти таблицы будут различаться, проще иметь две отдельные строки в двумерном массиве, чем во время вычислений отслеживать особый случай с февралем. Массив и функции, выполняющие преобразования, имеют следующий вид:

```
static char daytab[2][13] = {
        {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31},
        {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}
};

/* day_of_year: определяет день года по месяцу и дню */
int day_of_year(int year, int month, int day) {
        int i, leap;
        leap = year%4 == 0 && year%100 != 0 || year%400 == 0;
        for (i = 1; i < month; i++)
                day += daytab[leap][i];
        return day;
}

/* month_day: определяет месяц и день по дню года */
void month_day(int year, int yearday, int *pmonth, int *pday) {
        int i, leap;
        leap = year%4 == 0 && year%100 != 0 || year%400 == 0;
        for (i = 1; yearday > daytab[leap][i]; i++)
                yearday -= daytab[leap][i];
        *pmonth = i;
        *pday = yearday;
}
```

Напоминаем, что арифметическое значение логического выражения (например, выражения, с помощью которого вычислялось leap) равно либо нулю (ложь), либо единице (истина), так что мы можем использовать его как индекс в массиве daytab.

Массив daytab должен быть внешним по отношению к обеим функциям day_of _year и month_day, так как он нужен и той и другой. Мы сделали его типа char, чтобы проиллюстрировать законность применения типа char для малых целых без знака.

Массив daytab — это первый массив из числа двумерных, с которыми мы еще не имели дела. Строго говоря, в Си двумерный массив рассматривается как одномерный массив, каждый элемент которого — также массив. Поэтому индексирование изображается так:

```
daytab[i][j] /* [строка] [столбец] */
```

а не так:

```
daytab[i,j] /* НЕВЕРНО */
```

Особенность двумерного массива в Си заключается лишь в форме записи, в остальном его можно трактовать почти так же, как в других языках. Элементы запоминаются строками, следовательно, при переборе их в том порядке, как они расположены в памяти, чаще будет изменяться самый правый индекс.

Массив инициализируется списком начальных значений, заключенным в фигурные скобки; каждая строка двумерного массива инициализируется соответствующим подсписком. Нулевой столбец добавлен в начало daytab лишь для того, чтобы индексы, которыми мы будем пользоваться, совпадали с естественными номерами месяцев от 1 до 12. Экономить пару ячеек памяти здесь нет никакого смысла, а программа, в которой уже не надо корректировать индекс, выглядит более ясной.

Если двумерный массив передается функции в качестве аргумента, то объявление соответствующего ему параметра должно содержать количество столбцов; количество строк в данном случае несущественно, поскольку, как и прежде, функции будет передан указатель на массив строк, каждая из которых есть массив из 13 значений типа int, В нашем частном случае мы имеем указатель на объекты, являющиеся массивами из 13 значений типа int. Таким образом, если массив daytab передается некоторой функции f, то эту функцию можно было бы определить следующим образом:

```
f(int daytab[2][13]) {…}
```

Вместо этого можно записать

```
f(int daytab[][13]) {…}
```

поскольку число строк здесь не имеет значения, или

```
f(int (*daytab)[13]) {…}
```

Последняя запись объявляет, что параметр есть указатель на массив из 13 значений типа int. Скобки здесь необходимы, так как квадратные скобки [] имеют более высокий приоритет, чем *. Без скобок объявление int *daytab[13] определяет массив из 13 указателей на char. В более общем случае только первое измерение (соответствующее первому индексу) можно не задавать, все другие специфицировать необходимо.

Инициализация массивов указателей

Напишем функцию month_name(n), которая возвращает указатель на строку символов, содержащий название n-го месяца. Эта функция идеальна для демонстрации использования статического массива. Функция month_name имеет в своем личном распоряжении массив строк, на одну из которых она и возвращает указатель. Ниже покажем, как инициализируется этот массив имен.

```
/* month_name: возвращает имя n-го месяца */
char *month_name(int n) {
    static char *name[] = {
        "Неверный месяц", "Январь", "Февраль", "Март", "Апрель", "Май",
    "Июнь", "Июль", "Август", "Сентябрь", "Октябрь", "Ноябрь", "Декабрь"
```

```
        };
        return (n < 1 || n > 12) ? name[0] : name[n];
}
```

Объявление name массивом указателей на символы такое же, как и объявление lineptr в программе сортировки. Инициализатором служит список строк, каждой из которых соответствует определенное место в массиве. Символы i-й строки где-то размещены, и указатель на них запоминается в name[i]. Так как размер массива name не специфицирован, компилятор вычислит его по количеству заданных начальных значений.

Указатели против многомерных массивов

Начинающие программировать на Си иногда не понимают, в чем разница между двумерным массивом и массивом указателей вроде name из приведенного примера. Для двух следующих определений:

```
int a[10][20];
int *b[10];
```

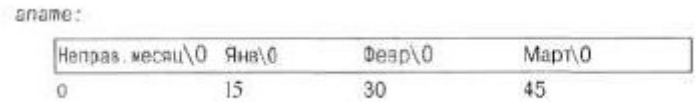
записи a[3][4] и b[3][4] будут синтаксически правильным обращением к некоторому значению типа int. Однако только a является истинно двумерным массивом: для двухсот элементов типа int будет выделена память, а вычисление смещения элемента a[строка][столбец] от начала массива будет вестись по формуле 20 x строка + столбец, учитывающей его прямоугольную природу. Для b же определено только 10 указателей, причем без инициализации. Инициализация должна задаваться явно — либо статически, либо в программе. Предположим, что каждый элемент b указывает на двадцатиэлементный массив, в результате где-то будут выделены пространство, в котором разместятся 200 значений типа int, и еще 10 ячеек для указателей. Важное преимущество массива указателей в том, что строки такого массива могут иметь разные длины. Таким образом, каждый элемент массива b не обязательно указывает на двадцатиэлементный вектор; один может указывать на два элемента, другой — на пятьдесят, а некоторые и вовсе могут ни на что не указывать.

Наши рассуждения здесь касались целых значений, однако чаще массивы указателей используются для работы со строками символов, различающимися по длине, как это было в функции month_name. Сравните определение массива указателей и соответствующий ему рисунок:

```
char *name[] = {"Неправильный месяц", "Янв", "Февр", "Март"};
```

с объявлением и рисунком для двумерного массива:

```
char name[][15] = {"Неправ. месяц", "Янв", "Февр", "Март"};
```



Используемые источники:

1. Язык программирования Си. Авторы: Б. Керниган, Д. Ритчи.
2. Онлайн компилятор языка Си: https://www.onlinegdb.com/online_c_compiler
3. Онлайн блок-схемы: https://app.diagrams.net

Задание:

При решении каждой задачи, необходимо создать блок-схемы. Решением работы должен служить файл-отчёт, в котором будут располагаться скриншоты кода, выполнения программ, а также сами блок-схемы.

1. Реализовать функцию, которая в числовой матрице будет менять местами два столбца, т. е. все элементы одного столбца поставит на соответствующие им позиции другого, а элементы второго переместит в первый.
2. Реализовать функцию, которая будет считать сумму элементов в каждом столбце матрицы и определять на выходе какой столбец содержит максимальную сумму.
3. Реализовать функцию, которая будет находить положительные элементы главной диагонали квадратной матрицы и заносить их в массив.
4. Реализовать функцию, которая будет находить индексы всех минимальных элементов матрицы и заносить их в массив.
5. Реализовать функцию, которая будет находить максимальный элемент каждого столбца матрицы, после чего заносить данный элемент в массив.
6. Реализовать функцию, которая будет изменять последовательность столбцов матрицы так, чтобы элементы их первой строки были отсортированы по возрастанию.
7. Реализовать функцию, которая будет находить максимальный элемент среди минимальных элементов столбцов матрицы.
8. Реализовать функцию, которая будет находить количество двузначных чисел в матрице.