МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Белгородский государственный технологический университет

им. В.Г. Шухова

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**РАСЧЕТНО-ГРАФИЧЕСКОЕ ЗАДАНИЕ**

По дисциплине «Основы программирования»

Тема: Использование подпрограмм с параметрами функционального типа и не типизованными параметрами.

Автор работы \_\_\_\_\_\_\_\_\_\_\_ Воскобойников И. С. ВТ-12

(подпись)

Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_ Брусенцева В.С.

(подпись)

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Белгород

2019 г.

**Использование указателей на функции и создание программы из нескольких файлов**

**Цель работы:** Получение навыков передачи функций функциям в

качестве параметров, раздельной компиляции файлов программы и

создания модульной программы

**Задания для подготовки к работе**

1. Изучить правила описания и использования указателей на функции в языке Си.
2. Разработать алгоритм и описать функцию для универсальной сортировки произвольного массива с произвольным базовым типом. Функции передается массив как нетипизованный параметр, его длина, размер элемента и логическая функция сравнения двух элементов массива.
3. С использованием этой функции решить следующую задачу. Имеется информация о студентах группы: Ф.И.О., результаты последней экзаменационной сессии. Требуется получить список студентов с указанием среднего балла по итогам сессии, упорядоченный по указанию пользователя либо лексикографически, либо по невозрастанию среднего балла.
4. Функцию универсальной сортировки и необходимые для нее программные объекты описать в отдельном модуле.
5. Подобрать наборы тестовых данных.

**Алгоритм укрупненными блоками**

Начало

Ввод массива students структур типа t\_student размера n

Вывод массива students размера n

ра

Конец

n = 5

Ввод sortMode

sortMode ==2

Сортировка массива students по невозрастанию среднего балла

Сортировка массива students лексикографически

-

+



**Описание структур данных**

#define **MARKS\_NUMBER** 3  
#define **NAME\_SIZE** 40  
  
**typedef struct** \_t\_student {  
 **char** surname[**NAME\_SIZE**];  
 **char** name[**NAME\_SIZE**];  
 **char** fathername[**NAME\_SIZE**];  
 **int** marks[**MARKS\_NUMBER**];  
 **float** averageMarks;  
} t\_student;

**Описание функций**

1.Спецификация функции inputStudentArr():

1. Заголовок: void inputStudentArr(t\_student \*student,int n, int nMarks);

2. Назначение: ввод полей surname, name, fathername и marks и заполнение поля

averageMarks переменной student.

Блок-схема:

inputStudentArr (students,n, nMarks)

Student[j].averageMarks = 0.0

Ввод student[j].srname, student[j].name, student[j].fathername

i= 0, nMarks

Ввод student[j].marks[i]

student[j].averageMarks= student[j].averageMarks + student[j].marks[i]

Student[j].averageMarks = student[j].averageMarks ÷ nMarks

Выход

j = 0, n

2.Спецификация функции outputStudentArr ():

1. Заголовок: void outputStudentArr(t\_student \*student,int n);

2. Назначение: вывод полей surname, name, fathername и average членов

массива student.

Блок-схема:

Для упрощения понимания бок-схемы будем считать, что мы имеем дело с разыменованной структурой типа Student.

outputStudentArr(student,n)

i = 0, n

Вывод student[i].srname, student[i].name, student[i].fathername,student[i]. averageMarks

Выход

3.Специя функции cmpMarks

1. Заголовок: int cmpMarks(const void \*p1, const void \*p2)

2. Назначение: возвращает -1, если значение p1.averageMarks больше либо равно значению p2.averageMarks иначе 1

Блок-схема:

cmpMarks(p1, p2)

f=результат сравнения p1. averageMarks и p2. averageMarks

f>0

Возврат -1

Возврат 1

4.Спецификация функции cmpNames():

1. Заголовок: int cmpNames(const void \*p1, const void \*p2);

2. Назначение: Возвращает положительное число, если ФИО p1 > p2, возвращает отрицательное, если p1 < p2, и если p1 и p2 равны, возвращает 0

Блок-схема:

cmpNames(p1, p2)

Возврат rez

rez=результат сравнения p1.surname и p2.surname

rez=результат сравнения

p1.name и p2.name

rez!=0

Возврат rez

rez!=0

Возврат результата сравнения

p1.fathername и p2.fathername

5. Спецификация функции swap():

1. Заголовок: void swap(void \*a, void \*b, size\_t size);

2. Назначение: обмен байтов количеством size, расположенных по указателю a, с байтами, расположенными по указателю b.

Блок-схема:

swap(a, b, size)

i = 0, size

tmp = ac[i];

ac[i] = bc[i]

bc[i] = tmp;

Выход

ac = a указатель на однобайтовый тип

bc = b указателm на однобайтовый тип

6. Спецификация функции sort():

1. Заголовок: void sort (void\* arr, size\_t n, size\_t size, int (\*cmp)(const void\* a, const void\* b))

2. Назначение: сортирует выбором массив arr размера n, элементы которого имеют размер size. Алгоритм сравнивает пары значений путем вызова функции cmp.

Блок-схема:

cmp(pj,pMin)<0

sort(arr, n, size, cmp)

i = 0, n

min = i

j=i+1, j<n

pi= arr, приведенный к однобайтовому типу

min = j

imin != i

swap(&pi[j],

pMin[j],sizeof(\*pi))

Выход

+

—

—

pMin = pi

pMin = pj

+

j=0,j<size

В программе используется модуль сортировки, который состоит из двух файлов:

* файл заголовок — UNIVERS\_SORT.h
* файл исходного кода — UNIVERS\_SORT.c

**Заголовок модуля сортировки:**

#ifndef **RGZPROGA\_UNIVERS\_SORT\_H**#define **RGZPROGA\_UNIVERS\_SORT\_H**#include **<stdint.h>  
  
  
void** swap(**void** \*a, **void** \*b, size\_t size);  
**void** sort( **void**\* array, size\_t n, size\_t width, **int** (\*cmp)(**const void**\* a, **const void**\* b) );  
  
#endif *//RGZPROGA\_UNIVERS\_SORT\_H*

**Модуль сортировки:**

#include **"UNIVERS\_SORT.h"**#include **<stdio.h>**#include **<stddef.h>  
  
void** swap(**void** \*a, **void** \*b, size\_t size) {  
 **char** tmp;  
 **char**\* ac = (**char**\*)(a);  
 **char**\* bc = (**char**\*)(b);  
 **for**( **int** i = 0; i < size; i++){  
 tmp = ac[i];  
 ac[i] = bc[i];  
 bc[i] = tmp;  
 }  
}  
  
**void** sort (**void**\* arr, size\_t n, size\_t size, **int** (\*cmp)(**const void**\* a, **const void**\* b)){  
 size\_t i, j, min;  
 **char** \*pMin, \*pi, \*pj;  
 pi = (**char**\*)arr;  
 **for**( i = 0; i < n-1; ++i, pi += size ){  
 pMin = pi;  
 min = i;  
 **for**( pj = pi+size, j = i+1; j < n; ++j, pj += size ){  
 **if**( cmp( pj, pMin ) < 0 ){  
 pMin = pj;  
 min = j;  
 }  
 }  
 **if**( min != i ){  
 **for**( j = 0; j < size; ++j ){  
 swap(&pi[j],&pMin[j], **sizeof**(\*pi));  
 }  
 }  
 }  
}

**Основная программа:**

#include **<stdio.h>**#include **<string.h>**#include **<malloc.h>**#include **"UNIVERS\_SORT.h"**#define **N** 5  
#define **MARKS\_NUMBER** 3  
#define **NAME\_SIZE** 40  
  
**typedef struct** \_t\_student {  
 **char** surname[**NAME\_SIZE**];  
 **char** name[**NAME\_SIZE**];  
 **char** fathername[**NAME\_SIZE**];  
 **int** marks[**MARKS\_NUMBER**];  
 **float** averageMarks;  
} t\_student;  
  
**void** inputStudentArr(t\_student \*student,**int** n, **int** nMarks){  
 **for** (**int** j = 0; j < n; ++j) {  
 student[j].averageMarks = 0;  
 scanf(**"%s%s%s"**, student[j].surname, student[j].name, student[j].fathername);  
  
 **for** (**int** i = 0; i < nMarks; ++i) {  
 scanf(**"%d"**, &student[j].marks[i]);  
 student[j].averageMarks += student[j].marks[i];  
 }  
 student[j].averageMarks = student[j].averageMarks / nMarks;  
 }  
}  
  
**void** outputStudentArr(t\_student \*student,**int** n){  
 **for** (**int** i = 0; i < n; ++i) {  
 printf(**"%s %s %s %f\n"**, student[i].surname,student[i].name,student[i].fathername,student[i].averageMarks);  
 }  
}  
  
**int** cmp\_Marks(**const void**\* a, **const void**\* b) {  
 **float** f = (((t\_student\*)a)->averageMarks )- (((t\_student\*)b)->averageMarks);  
 **if** (f > 0) {  
 **return** -1;  
 } **else** {  
 **return** 1;  
 }  
}  
  
**int** compNames(**const void** \*p1, **const void** \*p2){  
 **int** res = strcmp(((t\_student\*) p1)->surname, ((t\_student \*) p2)->surname);  
 **if**( res != 0){  
 **return** res;  
 } **else** {  
 res = strcmp(((t\_student \*) p1)->name, ((t\_student \*) p2)->name);  
 **if**( res != 0){  
 **return** res;  
 } **else** {  
 **return** strcmp(((t\_student \*) p1)->fathername, ((t\_student \*) p2)->fathername);  
 }  
 }  
}  
  
**int** main() {  
 **int** SortMode;  
 t\_student \*students = (t\_student\*)calloc(**N**, **sizeof**(t\_student));  
 inputStudentArr(students,**N**,**MARKS\_NUMBER**);  
 printf(**"Введите режим сортировки (1-по среднему балу, 2-по Фамилиям)\n"**);  
 scanf(**"%d"**,&SortMode);  
 **switch**( SortMode ){  
 **case** 1:  
 sort((**void**\*)students, **N**, **sizeof**(t\_student), cmp\_Marks);  
 outputStudentArr(students,**N**);  
 **break**;  
 **case** 2:  
 sort((**void**\*)students, **N**, **sizeof**(t\_student), compNames);  
 outputStudentArr(students,**N**);  
 **break**;  
 **default**:  
 printf(**"%s"**,**"Вы неправильно выбрали режим сортировки."**);  
 **break**;  
 }  
 free(students);  
 **return** 0;  
}

**Тестовые данные:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Иванкин Кирилл Сергеевич 5 4 5  Лампика Алиса Алексеевна 3 4 5  Артебякин Андрей Витальевич 4 4 5  Плешивцев Сергей Дмитриевич 5 5 5  Черных Артём Вячеславович 4 4 3  1 (по невозрастанию среднего балла) | Плешивцев Сергей Дмитриевич 5.000000  Иванкин Кирилл Сергеевич 4.666667  Артебякин Андрей Витальевич 4.333333  Лампика Алиса Алексеевна 4.000000  Черных Артём Вячеславович 3.666667 |
| Иванкин Кирилл Сергеевич 5 4 5  Лампика Алиса Алексеевна 3 4 5  Артебякин Андрей Витальевич 4 4 5  Плешивцев Сергей Дмитриевич 5 5 5  Черных Артём Вячеславович 4 4 3  2 (лексикографически) | Артебякин Андрей Витальевич 4.333333  Иванкин Кирилл Сергеевич 4.666667  Лампика Алиса Алексеевна 4.000000  Плешивцев Сергей Дмитриевич 5.000000  Черных Артём Вячеславович 3.666667 |
| Иванкин Кирилл Сергеевич 5 4 5  Лампика Алиса Алексеевна 3 4 5  Артебякин Андрей Витальевич 4 4 5  Плешивцев Сергей Дмитриевич 5 5 5  Черных Артём Вячеславович 4 4 3  33 | Вы неправильно выбрали режим сортировки. |