МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Белгородский государственный технологический университет имени В. Г. Шухова

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Рассчетно-графическая работа  
по дисциплине «Основы программирования» на тему:

«Использование подпрограмм с параметрами функционального типа и нетипизованными параметрами»

Выполнил:  
студент группы ПВ-11  
Адаменко И. И.

Проверил:  
старший преподаватель  
Гарибов А. И.

Белгород

2013

Оглавление

[Описание работы 3](#_Toc358854435)

[Задания для подготовки к работе 3](#_Toc358854436)

[Алгоритм работы программы 4](#_Toc358854437)

[Константы и используемые типы 5](#_Toc358854438)

[Описание подпрограмм модуля и библиотеки 6](#_Toc358854439)

[Спецификация процедуры swap 6](#_Toc358854440)

[Спецификация процедуры sort 7](#_Toc358854441)

[Описание подпрограмм используемых в программе 9](#_Toc358854442)

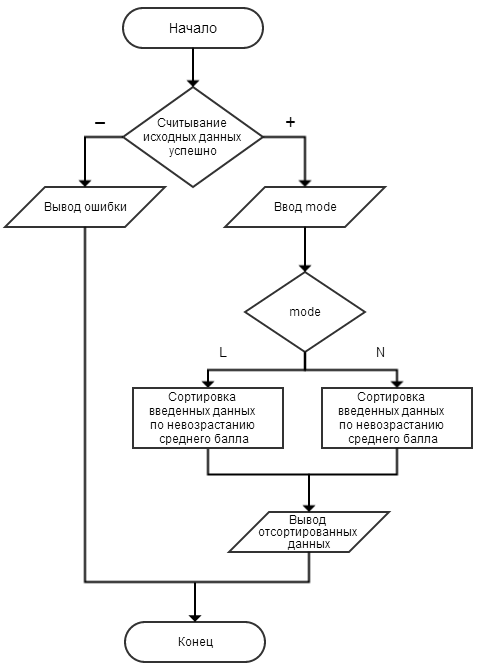
# Описание работы

Цель работы: получение навыков описания и использования объектов процедурного типа и нетипизованных параметров.

## Задания для подготовки к работе

1. Изучить структуры и объединения в языке Си.
2. Изучить процедурный и функциональный тип в Турбо Паскале и тип указатель на функцию в языке Си, организацию передачи подпрограмм подпрограммам в качестве параметров, приемы обработки нетипизованных параметров подпрограмм.
3. Разработать алгоритм и описать подпрограмму для универсальной сортировки произвольного массива с произвольным базовым типом. Подпрограмме передается массив как нетипизованный параметр, его длина, размер элемента и логическая функция сравнения двух элементов массива.
4. С использованием этой подпрограммы 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.
5. Программы создать на языках Паскаль и Си. Подпрограмму универсальной сортировки и необходимые для нее программные объекты описать в отдельном модуле.
6. Подобрать наборы тестовых данных.

# Алгоритм работы программы



# Константы и используемые типы

Паскаль

**type**

t\_func = function(**var** a, b): **boolean**;  
student = **record** name: **string**[50];  
 marks: **array** [1..marks\_number] of **byte**;  
 avg: **real**;  
end;  
pstudent = ^student;  
students = **array** [1..65520 div sizeof(student)] of **student**;  
pstudents = ^students;

**const**

marks\_number = 3;

Си

**#define** marks\_number 3

**typedef struct** {  
 **char** name[50];  
 **int** marks[marks\_number];  
 **float** avg;  
} student;

# Описание подпрограмм модуля и библиотеки

## Спецификация процедуры swap

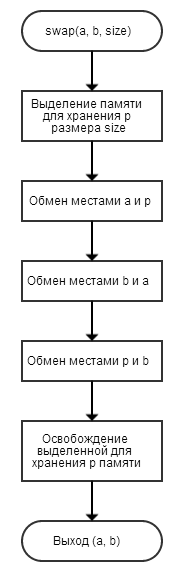
Паскаль:

1. Заголовок: **procedure** swap(**var** a, b; size: **byte**);
2. Назначение: меняет местами элементы a и b размера size.
3. Входные параметры: a, b, size.
4. Выходные параметры: a, b.

Си:

1. Заголовок: void swap(void \*a, void \*b, size\_t size);
2. Назначение: меняет местами элементы a и b размера size.

Блок-схема процедуры



## Спецификация процедуры sort

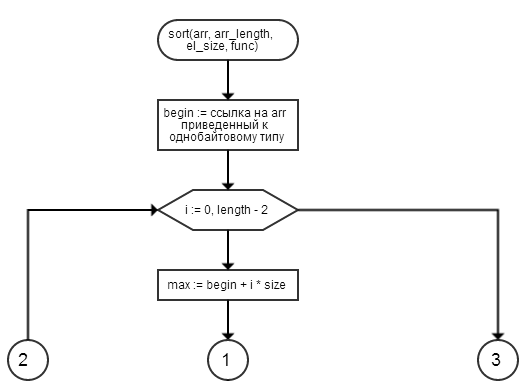
Паскаль:

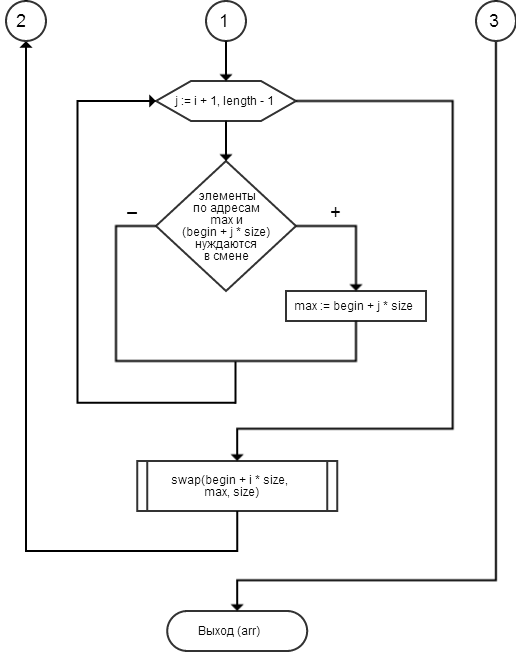
1. Заголовок: **procedure** sort(**var** arr; arr\_length, el\_size: **byte**; func: **t\_func**);
2. Назначение: сортирует массив arr произвольного типа, с количеством элементов arr\_length, каждый из которых размера el\_size, используя функцию func.
3. Входные параметры: arr, arr\_length, el\_size, func.
4. Выходные параметры: arr.

Си:

1. Заголовок: **void** sort(**void \*** arr, **unsigned** length, **size\_t** size, **int** (\*funct)(**void \***, **void \***));
2. Назначение: сортирует массив arr произвольного типа, с количеством элементов length, каждый из которых размера size, используя функцию funct.

Блок-схема процедуры





# Описание подпрограмм используемых в программе

## Спецификация функции readFile

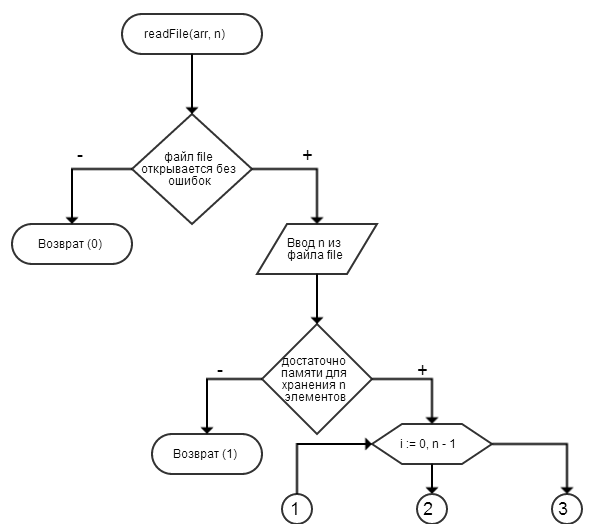
Паскаль:

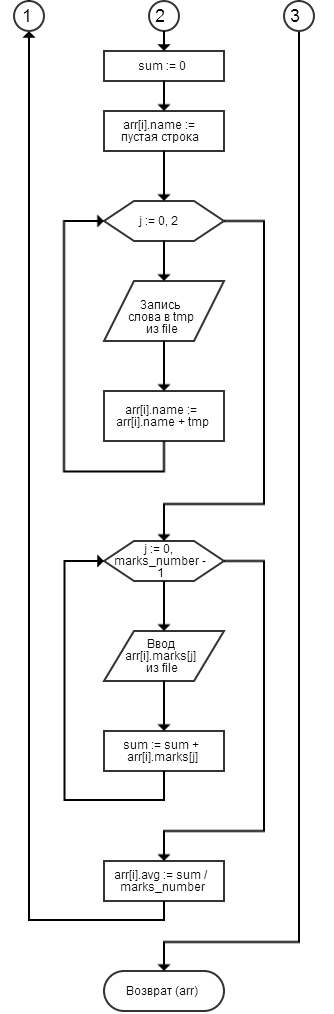
1. Заголовок: **function** readFile(**var** a: pstudents): **boolean**;
2. Назначение: ввод данных в массив a.
3. Входные параметры: a.
4. Выходные параметры: a.

Си:

1. Заголовок: **size\_t** readFile(student \*arr, **unsigned** \*n);
2. Назначение: ввод данных в массив **arr,** запись количества введенных элементов в **n.**

Блок-схема функции





## Спецификация процедуры writeFile

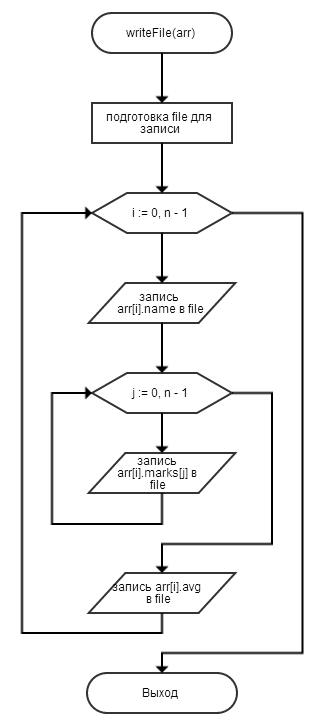
Паскаль:

1. Заголовок: **procedure** writeFile(arr: pstudents);
2. Назначение: вывод данных из массива arr в файл.
3. Входные параметры: arr.
4. Выходные параметры: нет.

Си:

1. Заголовок: **void** writeFile(student \*arr, **unsigned** n);
2. Назначение: вывод n элементов массива arr в файл.

Блок-схема функции



## Спецификация функции lexicographical

Паскаль:

1. Заголовок: **function** lexicographical(**var** a, b): **boolean**; **far**;
2. Назначение: возвращает TRUE, если a лексико-графически меньше, чем b, иначе — FALSE.
3. Входные параметры: a, b.
4. Выходные параметры: a, b.

Си:

1. Заголовок: **int** lexicographical(**void \***a, **void \***b);
2. Назначение: возвращает TRUE, если a лексико-графически меньше, чем b, иначе — FALSE.

Блок-схема функции



## Спецификация функции nonincreasing

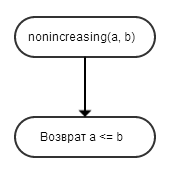
Паскаль:

1. Заголовок: **function** nonincreasing(**var** a, b): **boolean**; **far**;
2. Назначение: возвращает TRUE, если a меньше или равно b, иначе — FALSE.
3. Входные параметры: a, b.
4. Выходные параметры: a, b.

Си:

1. Заголовок: **int** nonincreasing(**void \***a, **void \***b);
2. Назначение: возвращает TRUE, если a меньше или равно b, иначе — FALSE.

Блок-схема функции



# Тестовые данные

|  |  |  |  |
| --- | --- | --- | --- |
| № | Исходные данные | | Результат работы программы |
| Содержание файла input.txt | mode | Содержание файла output.txt |
| 1 | 3  QQQ UUU FFF 3 2 5  ZZZ YYY XXX 5 5 5  AAA BBB CCC 5 4 2 | L | AAA BBB CCC 5 4 2 3.67  QQQ UUU FFF 3 2 5 3.33  ZZZ YYY XXX 5 5 5 5.00 |
| 2 | 3  QQQ UUU FFF 3 2 5  ZZZ YYY XXX 5 5 5  AAA BBB CCC 5 4 2 | N | ZZZ YYY XXX 5 5 5 5.00  AAA BBB CCC 5 4 2 3.67  QQQ UUU FFF 3 2 5 3.33 |

# Тексты программ

## Паскаль

Содержание файла модуля U\_SORT

**UNIT** U\_SORT;

{----------------------------------------------------------------}

**INTERFACE**

{----------------------------------------------------------------}

**type**

t\_func = **function**(**var** a, b): **boolean**;

**procedure** swap(**var** a, b; size: **byte**);

**procedure** sort(**var** a; a\_length, el\_size: **byte**; func: t\_func);

{----------------------------------------------------------------}

**IMPLEMENTATION**

{----------------------------------------------------------------}

**procedure** swap(**var** a, b; size: **byte**);

**var**

p: **pointer**;

**begin**

**getmem**(p, size);

**move**(a, p^, size);

**move**(b, a, size);

**move**(p^, b, size);

**freemem**(p, size);

**end**;

**procedure** sort(**var** arr; arr\_length, el\_size: **byte**;   
func: t\_func);

**type**

pbyte = ^**byte**;

**var**

start, first, second, max: pbyte;

i, j: **byte**;

**begin**

start := pbyte(arr);

**for** i := 0 **to** arr\_length - 2 **do**

**begin**

first := pbyte(**longword**(start) + i\*el\_size);

max := first;

**for** j := i + 1 **to** arr\_length - 1 **do**

**begin**

second := pbyte(**longword**(start) + j\*el\_size);

**if** (func(second^, max^)) **then**

max := second;

**end**;

swap(first^, max^, el\_size);

**end**;

**end**;

**begin**

*//nothing to do here*

**end**.

Содержание файла основной программы

**program** lab\_18;

*{$MODE TP}*

**uses** U\_SORT;

**const**

marks\_number = 3;

**type**

student = **record**

name: **string**[50];

marks: **array** [1..marks\_number] of **byte**;

avg: **real**;

end;

pstudent = ^student;

students = **array** [1..65520 div sizeof(student)] of student;

pstudents = ^students;

*{$F+}*

**function** lexicographical(**var** a, b): **boolean**; **far**;

**begin**

lexicographical := pstudent(@a)^.name < pstudent(@b)^.name;

**end**;

**function** nonincreasing(**var** a, b): **boolean**; **far**;

**begin**

nonincreasing := pstudent(@a)^.avg <= pstudent(@b)^.avg;

**end**;

*{$F-}*

**function** readFile(**var** arr: pstudents): **boolean**;

**var**

i, j, sum, counter: **byte**;

input: **text**;

c: **char**;

**begin**

readFile := **FALSE**;

**assign**(input, *'input.txt'*);

**reset**(input);

**ReturnNilIfGrowHeapFails** := **TRUE**;

**getmem**(arr, sizeof(student) \* n);

i := 1;

**if** (arr <> nil) **then** **begin**

**read**(input, n);

**while** (i <= n) **do**

**begin**

arr^[i].name := *''*;

counter := 0;

**while** (counter < 3) **do**

**begin**

**read**(input, c);

arr^[i].name := arr^[i].name +   
 c;

**if** c = *' '* **then**

**inc**(counter);

**end**;

j := 1;

sum := 0;

**while** (j <= marks\_number) **do**

**begin**

**read**(input, arr^[i].marks[j]);

sum := sum + arr^[i].marks[j];

**inc**(j);

**end**;

arr^[i].avg := sum / marks\_number;

**readln**(input);

**inc**(i);

**end**;

readFile := **TRUE**;

**end**;

**close**(input);

**end**;

**procedure** writeFile(arr: pstudents);

**var**

i, j: **byte**;

output: **text**;

**begin**

**assign**(output, 'output.txt');

**rewrite**(output);

**for** i := 1 **to** n **do**

**begin**

**write**(output, arr^[i].name);

**for** j := 1 **to** marks\_number **do**

**write**(output, arr^[i].marks[j], *' '*);

**writeln**(output, arr^[i].avg:3);

**end**;

**close**(output);

**end**;

**var**

arr: pstudents;

mode: **char**;

**begin**

**if** (readFile(arr)) **then**

**begin**

**writeln**(*'[L]exicographical or [N]onincreasing?'*);

**readln**(mode);

**case** (mode) **of**

'L': sort(arr, n, sizeof(student), lexicographical);

'N': sort(arr, n, sizeof(student), nonincreasing);

**else**

**writeln**(*'Error! Input of the initial data!'*);

**end**;

writeFile(arr);

**end**

**else**

**writeln**(*'Not enough memory!'*);

**readln**;

**end**.

## Си

Содержание файла библиотеки u\_sort

**#include** <stdio.h>

**#include** <stdlib.h>

**#include** "u\_sort.h"

**void** swap(**void \***a, **void \***b, **size\_t** size) {

**char** tmp;

**int** i = 0;

**for** (; i < size; i++) {

tmp = ((**char \***)a)[i];

((**char \***)a)[i] = ((**char \***)b)[i];

((**char \***)b)[i] = tmp;

}

}

**void** sort(**void \*** arr, **unsigned** length, **size\_t** size, **int** (\*funct)(**void \***, **void \***)) {

**char \***begin = (**char \***)arr, **\***max;

**int** i = 0, j;

**for** (;i < length - 1; i++) {

max = begin + i \* size;

**for** (j = i + 1; j < length; j++)

**if** (funct(begin + j \* size, max)) {

max = begin + j \* size;

}

swap(begin + i \* size, max, size);

}

}

Содержание файла основной программы

**#include** <stdio.h>

**#include** <stdlib.h>

**#include** "u\_sort.h"

**#define** marks\_number 3

**typedef struct** {

**char** name[50];

**int** marks[marks\_number];

**float** avg;

} student;

**int** lexicographical(**void \***a, **void \***b) {

**int** i = 0;

**for** (; (\*(student \*)a).name[i] && (\*(student \*)a).name[i] ==

(\*(student \*)b).name[i]; i++) ;

**return** ((\*(student \*)a).name[i] < (\*(student \*)b).name[i]);

}

**int** nonincreasing(**void \***a, **void \***b) {

**return** ((\*(student \*)a).avg <= (\*(student \*)b).avg);

}

**void** funcStrcat(**char \***to, **char \***from) {

**while**(\*to) to++;

**while**(\*to++ = \*from++);

\*(--to) = *' '*;

\*(++to) = *'\0'*;

}

**size\_t** readFile(student \*arr, **unsigned \***n) {

**size\_t** i = 0, j, sum;

**FILE** **\***file;

**char** tmp[20];

**if** (file = **fopen**(*"input.txt"*, *"r"*)) {

**fscanf**(file, *"%u\n"*, &\*n);

**if** (arr = (student \*) **malloc** (sizeof(student) \* \*n)) {

**for** (; i < \*n; i++) {

sum = 0;

arr[i].name[0] = *'\0'*;

**for** (j = 0; j < 3; j++) {

**fscanf**(file, *"%s"*, tmp);

funcStrcat(arr[i].name, tmp);

}

**for** (j = 0; j < marks\_number; j++) {

**fscanf**(file,*"%d"*, &arr[i].marks[j]);

sum += arr[i].marks[j];

}

arr[i].avg = (**float**)sum/marks\_number;

}

**return** (**size\_t**)arr;

} **else return** 1;

} **else return** 0;

}

**void** writeFile(student \*arr, **unsigned** n) {

**size\_t** i = 0, j;

**FILE \***file;

file = **fopen**(*"output.txt"*, *"w"*);

**for** (; i < n; i++) {

**fprintf**(file, *"%s"*, arr[i].name);

**for** (j = 0; j < n; j++)

**fprintf**(file, *"%d "*, arr[i].marks[j]);

**fprintf**(file, *"%.2f\n"*, arr[i].avg);

}

**fclose**(file);

}

**int** main() {

student \*arr = (student \*) **malloc** (sizeof(student));

**unsigned** n;

**size\_t** p;

**char** type;

**switch** (p = readFile(arr, &n)) {

**case** 0: **printf**(*"%s"*, *"File not opened."*); **break**;

**case** 1: **printf**(*"%s"*, *"Not enought memory."*); **break**;

**default**:

arr = (student \*)p;

**printf**(*"%s\n"*, *"[L]exicographical or*

*[N]onincreasing?"*);

**scanf**(*"%c"*, &type);

**switch** (type) {

**case** 'N':

sort(arr, n, sizeof(student), &nonincreasing);

**break**;

**case** 'L':

sort(arr, n, sizeof(student), &lexicographical);

**break**;

}

writeFile(arr, n);

**break**;

}

**return** 0;

}