

Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и технологий  
Высшая школа программной инженерии

# КУРСОВАЯ РАБОТА

**Лабораторная работа 1**  
по дисциплине: «Алгоритмы и структура данных»

Выполнил  
студент гр.в3530904/10022

Оберемок И.С.

Руководитель  
старший преподаватель.

Фёдоров С.А.

«23» мая 2022 г.

Санкт-Петербург  
2022

## Введение

Дан список группы в виде: Фамилия, Пол, Результаты сессии. Определить лидеров среди мужчин и женщин по успеваемости и их среди балл (у мужчин и женщин отдельно).

**Цель работы:** Одна и та же информация из входного файла вводится по-разному в оперативную память с целью освоения работы с различными структурами данных - задание выполняется в виде 5 отдельных программных проектов, где необходимо использовать разные способы обработки данных, то есть:

- массивы строк;
- массивы символов;
- внутренние процедуры;
- массивы структур или структур массивов;
- файлы записей;
- модули;
- хвостовая рекурсия;
- двонаправленные списки заранее неизвестной длины;
- регулярное программирование.

Реализовать задание с использованием массивов строк.

Реализовать задание с использованием массивов символов.

Реализовать задание с использованием массивов строк или строк массивов.

Реализовать задание с использованием динамической структуры данных.

# Глава 1

Данные на вход

Дудиков	Д. Р. М	43453	0.00
Тихонов	Л. П. М	55353	0.00
Степин	К. Д. М	55445	0.00
Садовникова	П. О. Ж	43555	0.00
Воробьева	Е. Р. Ж	44553	0.00

Данные на выход

Исходный список:

Дудиков	Д. Р. М	43453	0.00
Тихонов	Л. П. М	55353	0.00
Степин	К. Д. М	55445	0.00
Садовникова	П. О. Ж	43555	0.00
Воробьева	Е. Р. Ж	44553	0.00

Лучшая успеваемость среди юношей:

Степин	К. Д. М	55445	4.60
--------	---------	-------	------

Лучшая успеваемость среди девочек:

Садовникова	П. О. Ж	43555	4.40
-------------	---------	-------	------

Средний балл среди юношей: 4.20

Средний балл среди девочек: 4.30

## 1. Реализация задания с использованием массивов строк

Строковые массивы хранят части текста и обеспечивают набор функций для работы с текстом как данные. Можно индексировать в, измениться и конкатенировать массивы строк, как вы можете с массивами любого другого типа. Также можно получить доступ к символам в строке и добавить текст к строкам.

```
integer, parameter :: STUD_AMOUNT = 5, SURNAME_LEN = 15, &
INITIALS_LEN = 5, MARKS_AMOUNT = 5
character(kind=CH_), parameter :: MALE = Char(1052, CH_)
character(kind=CH_), parameter :: FEMALE = Char(1046, CH_)
integer : In, Out, IO, i
character(SURNAME_LEN, kind=CH_): Surnames(STUD_AMOUNT) = ""
```

```

character(SURNAME_LEN, kind=CH_), allocatable:: Boys_Surnames(:), &
                                     Girls_Surnames(:)
character(INITIALS_LEN, kind=CH_):: Initials(STUD_AMOUNT) = ""
character(INITIALS_LEN, kind=CH_), allocatable :: Boys_Initials(:), &
                                     Girls_Initials(:)

character(kind=CH_):: Gender(STUD_AMOUNT) = ""

open (file=input_file, encoding=E_, newunit=In)
    format = '(3(a, 1x), ' // MARKS_AMOUNT // 'i1, f5.2)'
    read (In, format, iostat=IO) (Surnames(i), Initials(i), &
        Gender(i), Marks(i, :), Aver_Marks(i), i = 1, STUD_AMOUNT)
    close (In)

do concurrent (i = 1:Boys_Amount)
    Boys_Surnames(i) = Surnames(Boys_Pos(i))
    Boys_Initials(i) = Initials(Boys_Pos(i))
    Boys_Marks(i, :) = Marks(Boys_Pos(i), :)
end do

do concurrent (i = 1:Girls_Amount)
    Girls_Surnames(i) = Surnames(Girls_Pos(i))
    Girls_Initials(i) = Initials(Girls_Pos(i))
    Girls_Marks(i, :) = Marks(Girls_Pos(i), :)
end do

```

## 2. Реализация задания с использованием массивов символов

Для получения регулярного доступа к оперативной памяти и сплюснутым данным в памяти, нужно назначить индексы в двумерном массиве символов (**Names(LEN, AMOUNT)**), тем самым считывая данные из списка по столбцам на не по строкам.

```

integer, parameter :: STUD_AMOUNT = 5, SURNAME_LEN = 15, &
INITIALS_LEN = 5, MARKS_AMOUNT = 5
character(kind=CH_), parameter:: MALE = Char(1052, CH_)
character(kind=CH_),parameter:: FEMALE = Char(1046, CH_)
character(kind=CH_):: Surnames(SURNAME_LEN, STUD_AMOUNT) = "", &
    Initials(INITIALS_LEN, STUD_AMOUNT) = "", &
    Genders(STUD_AMOUNT) = ""

character(kind=CH_), allocatable:: Boys_Surnames(:, :), Girls_Surnames(:, :)
character(kind=CH_), allocatable:: Boys_Initials(:, :), Girls_Initials(:, :)

integer:: Marks(MARKS_AMOUNT, STUD_AMOUNT) = 0, i = 0
integer, allocatable:: Boys_Marks(:, :), Girls_Marks(:, :)

subroutine Read_class_list(Input_File, Surnames, Initials, Genders, &
    Marks, Aver_Marks)
    character(*)
        Input_File

```

```

character(kind=CH_) Surnames(:, :), Initials(:, :), Genders(:)
integer              Marks(:, :)
real(R_)            Aver_Marks(:)
intent (in)         Input_File
intent (out)        Surnames, Initials, Genders, Marks, Aver_Marks

integer In, IO, i
character(:), allocatable :: format

open (file=Input_File, encoding=E_, newunit=In)
  format = '(' // SURNAME_LEN // 'a1, 1x, ' // INITIALS_LEN // &
    'a1, 1x, a, 1x, ' // MARKS_AMOUNT // 'i1, f5.2)'
  read (In, format, iostat=IO) (Surnames(:, i), Initials(:, i), &
    Genders(i), Marks(:, i), Aver_Marks(i), &
    i = 1, STUD_AMOUNT)
  call Handle_IO_status(IO, "reading class list")
close (In)
end subroutine Read_class_list

allocate (Gender_Surnames(SURNAME_LEN, Gender_Amount), &
  Gender_Initials(INITIALS_LEN, Gender_Amount), Gender_Marks(MARKS_AMOUNT, &
    Gender_Amount))

do concurrent (i = 1:Gender_Amount)
  Gender_Surnames(:, i) = Surnames(:, Gender_Pos(i))
  Gender_Initials(:, i) = Initials(:, Gender_Pos(i))
  Gender_Marks(:, i) = Marks(:, Gender_Pos(i))
end do

```

### 3. Реализация задания с использованием массив структур

```

function Read_class_list(File_Data) result(Group)
type(student)          Group(STUD_AMOUNT)
character(*), intent(in) :: File_Data

integer In, IO, recl

recl = ((SURNAME_LEN + INITIALS_LEN + 1)*CH_ + &
  MARKS_AMOUNT*I_ + R_) * STUD_AMOUNT
open (file=File_Data, form='unformatted', newunit=In, &
  access='direct', recl=recl)
  read (In, iostat=IO, rec=1) Group
  call Handle_IO_status(IO, "reading unformatted class list")
close (In)
end function Read_class_list

Boys = Pack(Group, Group%Sex == MALE)
Girls = Pack(Group, Group%Sex == FEMALE)

```

```

do concurrent (i = 1:Size(Boys))
Boys(i)%Aver_mark = Real(Sum(Boys(i)%Marks), R_) / MARKS_AMOUNT
end do

do concurrent (i = 1:Size(Girls))
Girls(i)%Aver_mark = Real(Sum(Girls(i)%Marks), R_) / MARKS_AMOUNT
end do

```

#### 4. Реализация задания с использованием структур массивов

```

function Read_class_list(Data_File) result(Group)
type(student)          Group
character(*), intent(in)  :: Data_File

integer In, IO

open (file=Data_File, form='unformatted', newunit=In, access='stream')
  read (In, iostat=IO) Group
  call Handle_IO_status(IO, "reading unformatted class list")
close (In)
end function Read_class_list

call Output_class_list(output_file, Group, INDEXES, "Исходный список:", "rewi

do concurrent (i = 1:Size(Group%Aver_mark))
Group%Aver_mark(i) = Real(Sum(Group%Marks(:,i)), R_) / MARKS_AMOUNT
end do

```

#### 5. Реализация задания с использованием динамического списка

```

function Read_class_list(Input_File) result(Class_List)
type(student), pointer  :: Class_List
character(*), intent(in)  :: Input_File
integer In

open (file=Input_File, encoding=E_, newunit=In)
  Class_List => Read_student(In)
close (In)
end function Read_class_list

recursive function Read_student(In) result(Stud)
type(student), pointer  :: Stud
integer, intent(in)  :: In
integer IO
character(:), allocatable  :: format

allocate (Stud)

format = '(3(a, 1x), ' // MARKS_AMOUNT // 'i1, f5.2)'

```

```

    read (In, format, iostat=IO) stud%Surname, &
        stud%Initials, stud%Sex, stud%Marks, stud%Aver_Mark
    call Handle_IO_status(IO, "reading line from file")
    if (IO == 0) then
        Stud%next => Read_student(In)
    else
        deallocate (Stud)
        nullify (Stud)
    end if
end function Read_student

Group_List => Read_class_list(input_file)

if (Associated(Group_List)) then
    call Output_class_list(output_file, Group_List, 0.0, .TRUE., .TRUE., &
        "Исходный список:", "rewind")
    call Max_and_sum (Group_List, Boys_Max_Value, Girls_Max_Value, &
        Sum_Boys_Amount, Sum_Girls_Amount, Number_of_Boys, Number_of_Girls)
    Boys_Average_Marks = Sum_Boys_Amount / Number_of_Boys
    Girls_Average_Marks = Sum_Girls_Amount / Number_of_Girls
end if

```

## 6. Основные операторы обработки данных.

- Open
- Write
- Do concurrent
- If, Elseif, Else
- Type

7. При использовании стандартных функций в Fortran код всегда будет векторизоваться. Данные будут векторизоваться, если однопоточные приложения, выполняющие одну операцию в каждый момент времени, модифицируются для выполнения нескольких однотипных операций одновременно.

8. При выборе использования массивов структур или структур массивов, нужно исходить из самого задания, что вам нужно получить и какие данные вам нужно обработать. При использовании массива структур, а не структур массивов, когда алгоритм подразумевает работу с данными конкретной структуры, а не со множеством всех элементов вообще.

## 9. Динамическая структура.

### 9.1. Ссылки и адресаты

- i. Ссылка - переменная, связанная с другой переменной, называемой адресатом. Обращении к ссылке будет происходить обращение к адресату и наоборот.
- ii. Ссылки позволяют создавать динамические структуры данных - списки, деревья, очереди.

- iii. Все изменения происходящие с адресатом, дублируется в ссылке.
  - iv. Объект называется недостижимым если на него нельзя сослаться.
10. Формирование двоичного файла. Двоичный файл создаётся для производительности, так же конфигурационный файл формируется в двоичном файле. Чтобы работать с двоичными данными, нужно знать формат записи.

```

subroutine Create_data_file(Input_File, Data_File)
  character(*), intent(in)    :: Input_File, data_file

  type(student)               :: stud
  integer                     :: In, Out, IO, i, recl
  character(:), allocatable   :: format

  open (file=Input_File, encoding=E_, newunit=In)
  recl = (SURNAME_LEN + INITIALS_LEN + 1)*CH_ + MARKS_AMOUNT*I_ + R_
  open (file=Data_File, form='unformatted', &
    newunit=Out, access='direct', recl=recl)
    format = '(3(a, 1x), ' // MARKS_AMOUNT // 'i1, f5.2)'
    do i = 1, STUD_AMOUNT
      read (In, format, iostat=IO) stud
      call Handle_IO_status(IO, "reading &
        formatted class list, line " // i)
      write (Out, iostat=IO, rec=i) stud
      call Handle_IO_status(IO, "creating unformatted &
        file with class list, record " // i)
    end do
  close (In)
  close (Out)
end subroutine Create_data_file

character(:), allocatable   :: input_file, output_file, data_file

input_file = "../data/class.txt"
output_file = "output.txt"
data_file = "class.dat"

call Create_data_file(input_file, data_file)

Group = Read_class_list(data_file)

```



# Глава 2

## 1. Сравнение реализаций

### 1.1. Сравнительная таблица массива структур, структур массивов, динамический список

реализация	1.3	1.4	1.5
сплошные данные	-	+	-
регулярный доступ	-	+	+
векторизация	-	+	-
понециальная векторизация	-	+	-

### 1.2. При выборе структур массива данные в оперативной памяти будут сплошными, так же будет регулярный доступ к памяти и следовательно возможна реализация векторизации.

## 2. Вывод

При выполнении лабораторной работы 1, были использованы и реализованы несколько способов реализации структуры данных. Для решения задачи, было выбрана структура массивов так как при её использовании данные в оперативной памяти будут сплошными, так же будет регулярный доступ к памяти и следовательно возможна реализация векторизации, тем самым мы увеличиваем производительность при обработке данных.