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

КУРСОВАЯ РАБОТА

Разработка структур данных
по дисциплине: «Алгоритмы и Структуры Данных»

Выполнил
студент гр. 30030/2х В.Ю.Сподынейко

Руководитель
ст. преп. ВШПИ ИКНК С.А.Фёдоров

Санкт-Петербург
2023

Содержание

1 Реализация и анализ применения различных структур данных 4
1.1 Массив строк 4
1.2 Массив символов 6
1.3 Массив структур 9
1.4 Структура массивов 11
1.5 Массив структур с использованием хвостовой рекурсии при обработке данных 13
1.6 Структура массивов с использованием хвостовой рекурсии при обработке данных 15
1.7 Динамический однонаправленный список 17

2 Сравнение реализаций 19

3 Заключение 20

Задание

Вариант 18

Дан список группы в виде:

ФАМИЛИЯ	И. О.	ПОЛ	ГОД РОЖДЕНИЯ
15 симв.	5 симв.	1 симв.	1 симв.

Пример входного файла:

Иванов	И. Л.	М	1985
Петрова	Д. О.	Ж	1983

Сформировать отсортированные по убыванию возраста списки мужчин и женщин. Использовать для сортировки метод “выбором”.

Пример выходного файла:

Мужчины:

Иванов	И. Л.	М	1985

Женщины:

Петрова	Д. О.	Ж	1983

Введение

Цель работы – Выбор структуры данных для решения поставленной задачи на современных микроархитектурах.

Задачи:

1. Реализовать задание с использованием массивов строк.
2. Реализовать задание с использованием массивов символов.
3. Реализовать задание с использованием массивов структур.
4. Реализовать задание с использованием структур массивов.
5. Реализовать задание с использованием массивов структур или структур массивов (на выбор) и с использованием хвостовой рекурсии при обработке данных.
6. Реализовать задание с использованием динамического списка.
7. Провести анализ на регулярный доступ к памяти.
8. Провести анализ на векторизацию кода.
9. Провести сравнительный анализ реализаций.

1 Реализация и анализ применения различных структур данных

Исходный сортируемый список состоит из 259 000 объектов.

1.1 Массив строк

Данные в памяти сплошные. Регулярный доступ к памяти осуществляется при разбиении списка по полу[1]. При сортировке[2] доступ к памяти - нерегулярный.

Код векторизован при разбиении списка по полу[3] и при перестановке элементов[4] (с помощью векторного индекса). На других участках код не векторизуется или векторизация затруднена, так как имеются зависимости "чтение после записи"и условные ветвления.

Время обработки данных: 341.915с

Объявление струкутуры данных:

```
1  !____________________Strings_array_declaration_____________________!
2  implicit none
3  character(:), allocatable                    :: input_file, output_file
4
5  integer                                      :: STUD_AMOUNT
6  integer, parameter                           :: SURNAME_LEN  = 15
7  integer, parameter                           :: INITIALS_LEN = 5
8  character(kind=CH_), parameter               :: MALE = Char(1052, CH_), &
9                                                  FEMALE = Char(1046, CH_)
10
11 character(SURNAME_LEN,kind=CH_),allocatable  :: Surnames(:), &
12                                                 Boys_Surnames(:), &
13                                                 Girls_Surnames(:)
14 character(INITIALS_LEN,kind=CH_),allocatable:: Initials(:), &
15                                                 Boys_Initials(:), &
16                                                 Girls_Initials(:)
17 character(kind=CH_), allocatable              :: Genders(:)
18 integer, allocatable                          :: YOB(:), Boys_YOB(:), &
19                                                 Girls_YOB(:)
20 integer                                       :: i
```

[1] см. строки 2-12
[2] см. строки 15-43
[3] см. строки 2-12
[4] см. строки 24-26

Основные операторы обработки данных:

```
!____________________Getting_students_list_by_gender____________!
! Gender Logical mask
Is_A_Gender  = Genders == Gender
Gender_Amount = Count(Is_A_Gender)

! Gender arrays
allocate (Gender_Surnames(Gender_Amount), &
     Gender_Initials(Gender_Amount), Gender_YOB(Gender_Amount))

Gender_Surnames = Pack(Surnames, Is_A_Gender)
Gender_Initials = Pack(Initials, Is_A_Gender)
Gender_YOB      = Pack(YOB, Is_A_Gender)

!_________________Selection_sort_by_Year_Of_Birth_and_Surname______!
do j = 1, Size(YOB) - 1
   minInd = j
   do k = j + 1, Size(YOB)
      if (Swap(YOB, Surnames, Initials, k, minInd)) then
         minInd = k
      end if
   end do
   if (minInd /= j) then
      ! Swap two elements
      Surnames([j, minInd]) = Surnames([minInd, j])
      Initials([j, minInd]) = Initials([minInd, j])
      YOB([j, minInd]) = YOB([minInd, j])
   end if
end do

!_________________Two_elemets_comparison_by_all_fields____________!
Swap = .false.

if (YOB(i) < YOB(j)) then
   Swap = .true.
else if (YOB(i) == YOB(j)) then
   if (Surnames(i) < Surnames(j)) then
      Swap = .true.
   else if (Surnames(i) == Surnames(j)) then
      if (Initials(i) < Initials(j)) then
         Swap = .true.
      end if
   end if
end if
```

1.2 Массив символов

Данные в памяти сплошные. Регулярный доступ к памяти осуществляется при разбиении списка по полу[5] и при обращении к массивам символов во время сортировки[6].

Код векторизован на строках 3-8, 14-15, 37 и при перестановке двух элементов массива данных[7]. На других участках код не векторизуется так как имеются зависимости "чтение после записи"и условные ветвления.

Хранение исходных массивов символов осуществляется по строкам.
Исходные массивы:
Surnames(STUD_AMOUNT, SURNAME_LEN) и **Initials**(STUD_AMOUNT, SURNAME_LEN)
Расположение в памяти:
Surnames(SURNAME_LEN, STUD_AMOUNT) и **Initials**(SURNAME_LEN, STUD_AMOUNT)

Это позволяет обеспечивать регулярный доступ к памяти при сравнении двух строк исходного массива - они располагаются в памяти непрерывно (сплошные данные). Обход массивов в памяти осуществляется по столбцам: Surnames(:, j) и Initials(:, j).

Время обработки данных: 144.414с

Объявление струкутуры данных:

```
1  !____________________Symbols_array_declaration_____________________!
2  implicit none
3  character(:), allocatable          :: input_file, output_file
4
5  integer                            :: STUD_AMOUNT
6  integer, parameter                 :: SURNAME_LEN  = 15
7  integer, parameter                 :: INITIALS_LEN = 5
8  character(kind=CH_), parameter     :: MALE = Char(1052, CH_), &
9                                        FEMALE = Char(1046, CH_)
10 ! The matrixes (Surnames(:,:), Initials(:,:)) data stored in lines:
11 ! original          : Matrix(m, n)
12 ! stored and indexed: Matrix(n, m)
13 character(kind=CH_), allocatable :: Surnames(:, :), &
14                                       Boys_Surnames(:, :), &
15                                       Girls_Surnames(:, :)
16 character(kind=CH_), allocatable :: Initials(:, :), &
17                                       Boys_Initials(:, :), &
18                                       Girls_Initials(:, :)
19 character(kind=CH_), allocatable :: Genders(:)
20 integer, allocatable             :: YOB(:), Boys_YOB(:), Girls_YOB(:)
21
22 integer, allocatable             :: INDEXES(:), Gender_Pos(:)
23 integer                          :: Gender_Amount, i
```

[5] см. строки 3-17
[6] см. строки 35-38
[7] см. строки 52-60

Основные операторы обработки данных:

```
!____________________Getting_students_list_by_gender____________!
! Logical mask corresponding to the gender
Is_A_Gender  = Genders == Gender
Gender_Amount = Count(Is_A_Gender)

! Gender arrays
INDEXES = [(i, i = 1, STUD_AMOUNT)]
Gender_Pos = Pack(INDEXES, Is_A_Gender)
allocate (Gender_Surnames(SURNAME_LEN, Gender_Amount), &
     Gender_Initials(INITIALS_LEN, Gender_Amount), &
     Gender_YOB(Gender_Amount))
! Data matrix for a gender.
do concurrent (i = 1:Gender_Amount)
   Gender_Surnames(:, i)  = Surnames(:, Gender_Pos(i))
   Gender_Initials(:, i)  = Initials(:, Gender_Pos(i))
   Gender_YOB(i)  = YOB(Gender_Pos(i))
end do

!_____________Selection_sort_by_Year_Of_Birth_and_Surname_____!
do j = 1, Size(YOB) - 1
   minInd = j
   do k = j + 1, Size(YOB)
      if (Swap(YOB, Surnames, Initials, k, minInd)) then
         minInd = k
      end if
   end do
   call Swap_elements(Surnames, Initials, YOB, j, minInd)
end do

!_____________Two_elemets_comparison_by_all_fields____________!
Swap = .false.
if (YOB(i) < YOB(j)) then
   Swap = .true.
else if (YOB(i) == YOB(j)) then
   if (GT(Surnames(:, j), Surnames(:, i))) then
      Swap = .true.
   else if (ALL(Surnames(:, j) == Surnames(:, i)) &
        .and. GT(Initials(:, j), Initials(:, i))) then
      Swap = .true.
   end if
end if

!_____________Compare_two_character_arrays____________________!
! Searching the first differing symbol or stop at the last one
do  i = 1, Min(Size(arr1), Size(arr2)) - 1
   if (arr1(i) /= arr2(i)) &
        exit
end do
GT = arr1(i) > arr2(i)
```

```
50
51 !_________________Swap_two_elements_________________________________!
52 tmpSurname  = Surnames(:, i)
53 Surnames(:, i) = Surnames(:, j)
54 Surnames(:, j) = tmpSurname
55
56 tmpInitials = Initials(:, i)
57 Initials(:, i) = Initials(:, j)
58 Initials(:, j) = tmpInitials
59
60 YOB([i, j]) = YOB([j, i])
```

1.3 Массив структур

Данные в памяти не сплошные. Доступ к памяти нерегулярный.

Код векторизован при разбиении списка по полу[8] и при перестановке двух элементов списка[9]. На других участках код не векторизуется так как имеются зависимости "чтение после записи"и условные ветвления.

Время обработки данных: 74.358с

Объявление струкутуры данных:

```
1  !____________________Symbols_array_declaration___________________!
2  implicit none
3  character(:), allocatable              :: input_file, output_file, &
4                                            data_file
5
6  integer                                :: STUD_AMOUNT
7  integer, parameter                     :: SURNAME_LEN   = 15
8  integer, parameter                     :: INITIALS_LEN  = 5
9  character(kind=CH_), parameter         :: MALE = Char(1052, CH_), &
10                                           FEMALE = Char(1046, CH_)
11
12 type student
13    character(SURNAME_LEN, kind=CH_)    :: Surname  = ""
14    character(INITIALS_LEN, kind=CH_)   :: Initials = ""
15    character(kind=CH_)                 :: Gender   = ""
16    integer(I_)                         :: YOB      = 0
17 end type student
18
19 type(student), allocatable              :: Group(:), Boys(:), &
20                                           Girls(:)
```

[8] см. строки 2-3

[9] см. строку 13

Основные операторы обработки данных:

```
!____________________Getting_students_list_by_gender_____________!
Boys  = Pack(Group, Group%Gender == MALE)
Girls = Pack(Group, Group%Gender == FEMALE)

!_________________Selection_sort_by_Year_Of_Birth_and_Surname______!
do j = 1, Size(Group) - 1
   youngest_stud = j
   do k = j + 1, Size(Group)
      if (Swap(Group, k, youngest_stud)) then
         youngest_stud = k
      end if
   end do
   Group([j, youngest_stud]) = Group([youngest_stud, j])
end do

!_________________Two_elemets_comparison_by_all_fields____________!
Swap = .false.
if (Group(i)%YOB < Group(j)%YOB) then
   Swap = .true.
else if (Group(i)%YOB == Group(j)%YOB) then
   if (Group(i)%Surname < Group(j)%Surname) then
      Swap = .true.
   else if (Group(i)%Surname==Group(j)%Surname .and. &
        Group(i)%Initials < Group(j)%Initials) then
      Swap = .true.
   end if
end if
```

1.4 Структура массивов

Данные в памяти сплошные. Регулярный доступ к памяти осуществляется при разбиении списка по полу[10]. При сортировке[11] доступ к памяти - нерегулярный.

Код векторизован при разбиении списка по полу[12] и при перестановке двух элементов массива с помощью векторного индекса[13]

Время обработки данных: 61.151с

Объявление струкутуры данных:

```
1  !_________________Structure_of_arrays_declaration_________________!
2  implicit none
3  character(:), allocatable            :: input_file, output_file, &
4                                          data_file
5  integer                              :: STUD_AMOUNT
6  integer, parameter                   :: SURNAME_LEN = 15
7  integer, parameter                   :: INITIALS_LEN  = 5
8  character(kind=CH_), parameter       :: MALE = Char(1052, CH_), &
9                                          FEMALE = Char(1046, CH_)
10
11 type students
12    character(SURNAME_LEN,kind=CH_),allocatable  :: Surnames(:)
13    character(INITIALS_LEN,kind=CH_),allocatable :: Initials(:)
14    character(kind=CH_), allocatable             :: Genders(:)
15    integer(I_), allocatable                     :: YOBs(:)
16 end type students
17
18 type(students) :: Group, Male_Students, Female_Students
```

[10] см. строки 2-8
[11] см. строки 11-16 и 26-35
[12] см. строки 2-8
[13] см. строки 19-22

Основные операторы обработки данных:

```
!____________________Getting_students_list_by_gender____________!
Is_A_Gender = Group%Genders == Gender
genderAmount = Count(Is_A_Gender)

Sub_group%Surnames = Pack(Group%Surnames, Is_a_gender)
Sub_group%Initials = Pack(Group%Initials, Is_a_gender)
Sub_group%Genders  = Pack(Group%Genders, Is_a_gender)
Sub_group%YOBs     = Pack(Group%YOBs, Is_a_gender)

!_________________Selection_sort_by_Year_Of_Birth_and_Surname______!
do j = 1, Size(Group%Surnames) - 1
   youngest_stud = j
   do k = j + 1, Size(Group%Surnames)
      if (Swap(Group, k, youngest_stud)) then
         youngest_stud = k
      end if
   end do
   ! Swap two elements
   Group%Surnames([j,youngest_stud])=Group%Surnames([youngest_stud,j])
   Group%Initials([j,youngest_stud])=Group%Initials([youngest_stud,j])
   Group%Genders([j,youngest_stud]) =Group%Genders([youngest_stud,j])
   Group%YOBs([j,youngest_stud])    =Group%YOBs([youngest_stud,j])
end do

!_________________Two_elemets_comparison_by_all_fields____________!
Swap = .false.
if ((Group%YOBs(i)) < (Group%YOBs(j))) then
   Swap = .true.
else if (Group%YOBs(i) == Group%YOBs(j)) then
   if (Group%Surnames(i) < Group%Surnames(j)) then
      Swap = .true.
   else if (Group%Surnames(i) == Group%Surnames(j) .and. &
        Group%Initials(i) < Group%Initials(j)) then
      Swap = .true.
   end if
end if
```

1.5 Массив структур с использованием хвостовой рекурсии при обработке данных

Данные в памяти не сплошные. Доступ к памяти нерегулярный.

Код векторизован при разбиении списка по полу[14] и при перестановке двух элементов массива структур[15].

Время обработки данных: 572.333с

Объявление струкутуры данных:

```
1  !____________Array_of_structures_recursive_declaration___________!
2  implicit none
3  character(:), allocatable             :: input_file, output_file, &
4                                           data_file
5
6  integer                               :: STUD_AMOUNT
7  integer, parameter                    :: SURNAME_LEN   = 15
8  integer, parameter                    :: INITIALS_LEN  = 5
9  character(kind=CH_), parameter        :: MALE = Char(1052, CH_), &
10                                          FEMALE = Char(1046, CH_)
11
12 type student
13    character(SURNAME_LEN, kind=CH_)   :: Surname  = ""
14    character(INITIALS_LEN, kind=CH_)  :: Initials = ""
15    character(kind=CH_)                :: Gender   = ""
16    integer(I_)                        :: YOB      = 0
17 end type student
18
19 type(student), allocatable             :: Group(:), Boys(:), &
20                                          Girls(:)
```

[14] см. строки 2-3

[15] см. строку 8

Основные операторы обработки данных:

```
!____________________Getting_students_list_by_gender____________!
Boys  = Pack(Group, Group%Gender == MALE)
Girls = Pack(Group, Group%Gender == FEMALE)

!______Recursive_Selection_sort_by_Year_Of_Birth_and_Surname_____!
youngest_stud = Find_youngest(Group, N, 1, N)
! Put the younges at the array's tail position
Group([N, youngest_stud]) = Group([youngest_stud, N])

if (N >= 2) &
     call Sort_class_list(Group, N-1)

!_____________________Find_the_youngest_person__________________!
if (j == N) then
   res = youngest
else if (Swap(Group, j, youngest)) then
   res = Find_youngest(Group, j, j+1, N)
else
   res = Find_youngest(Group, youngest, j+1, N)
end if

!_________________Two_elemets_comparison_by_all_fields___________!
Swap = .false.
if (Group(i)%YOB > Group(j)%YOB) then
   Swap = .true.
else if (Group(i)%YOB == Group(j)%YOB) then
   if (Group(i)%Surname > Group(j)%Surname) then
      Swap = .true.
   else if (Group(i)%Surname==Group(j)%Surname .and. &
        Group(i)%Initials > Group(j)%Initials) then
      Swap = .true.
   end if
end if
```

1.6 Структура массивов с использованием хвостовой рекурсии при обработке данных

Данные в памяти сплошные. Регулярный доступ к памяти осуществляется при разбиении списка по полу[16]. При сортировке[17] доступ к памяти - нерегулярный.

Код векторизован при разбиении списка по полу[18] и при перестановке двух элементов массива с помощью векторного индекса[19]

Время обработки данных: 183.642c

Объявление струкутуры данных:

```
1  !____________Array_of_structures_recursive_declaration___________!
2  implicit none
3  character(:), allocatable              :: input_file, output_file, &
4                                            data_file
5  integer                                :: STUD_AMOUNT
6  integer, parameter                     :: SURNAME_LEN   = 15
7  integer, parameter                     :: INITIALS_LEN  = 5
8  character(kind=CH_), parameter         :: MALE = Char(1052, CH_), &
9                                            FEMALE = Char(1046, CH_)
10
11 type students
12    character(SURNAME_LEN, kind=CH_), allocatable  :: Surnames(:)
13    character(INITIALS_LEN, kind=CH_), allocatable :: Initials(:)
14    character(kind=CH_), allocatable               :: Genders(:)
15 end type students
16
17 type(students)                          :: Group, Male_Students, &
18                                            Female_Students
```

[16] см. строки 2-8
[17] см. строки 11-19 и 22-41
[18] см. строки 2-8
[19] см. строки 13-16

Основные операторы обработки данных:

```
!____________________Getting_students_list_by_gender____________!
Is_A_Gender = Group%Genders == Gender
genderAmount = Count(Is_A_Gender)

Sub_group%Surnames = Pack(Group%Surnames, Is_a_gender)
Sub_group%Initials = Pack(Group%Initials, Is_a_gender)
Sub_group%Genders = Pack(Group%Genders, Is_a_gender)
Sub_group%YOBs = Pack(Group%YOBs, Is_a_gender)

!______Recursive_Selection_sort_by_Year_Of_Birth_and_Surname_____!
youngest_stud = Find_youngest(Group, N, 1, N)
! Put the younges at the array's tail position
Group%Surnames([N, youngest_stud]) = Group%Surnames([youngest_stud, N])
Group%Initials([N, youngest_stud]) = Group%Initials([youngest_stud, N])
Group%Genders([N, youngest_stud])  = Group%Genders([youngest_stud, N])
Group%YOBs([N, youngest_stud])     = Group%YOBs([youngest_stud, N])

if (N >= 2) &
     call Sort_class_list(Group, N-1)

!_____________________Find_the_youngest_person_________________!
if (j == N) then
   res = youngest
else if (Swap(Group, j, youngest)) then
   res = Find_youngest(Group, j, j+1, N)
else
   res = Find_youngest(Group, youngest, j+1, N)
end if

!_________________Two_elemets_comparison_by_all_fields___________!
Swap = .false.
if ((Group%YOBs(i)) > (Group%YOBs(j))) then
   Swap = .true.
else if (Group%YOBs(i) == Group%YOBs(j)) then
   if (Group%Surnames(i) > Group%Surnames(j)) then
      Swap = .true.
   else if (Group%Surnames(i) == Group%Surnames(j) .and. &
        Group%Initials(i) < Group%Initials(j)) then
      Swap = .true.
   end if
end if
```

1.7 Динамический однонаправленный список

Данные в памяти не сплошные. Доступ к памяти нерегулярный.
Код не векторизуется.
Время обработки данных: 161.108с

Объявление струкутуры данных:

```
!_________________Dynamic_list_declaration_________________________!
implicit none
character(:), allocatable              :: input_file, output_file
character(kind=CH_), parameter         :: MALE = Char(1052, CH_), &
                                          FEMALE = Char(1046, CH_)
integer, parameter                     :: SURNAME_LEN   = 15
integer, parameter                     :: INITIALS_LEN  = 5
integer, parameter                     :: MARKS_AMOUNT  = 5

type student
   character(SURNAME_LEN, kind=CH_)    :: Surname  = ""
   character(INITIALS_LEN, kind=CH_)   :: Initials = ""
   character(kind=CH_)                 :: Gender   = ""
   Integer(R_)                         :: YOB      = 0
   type(student), pointer              :: next => Null()
end type student

type(student), pointer                 :: Group_List => Null(), &
                                          Boys_List => Null(), &
                                          Girls_List => Null()
```

Основные операторы обработки данных:

```
!____________________Getting_students_list_by_gender_____________!
if (Stud%Gender == ProcessedGenger) then
   allocate (List, source=Stud)
   List%next => Null()

   if (Associated(Stud%next)) &
        call Get_list_by_gender(Stud%next, List%next, ProcessedGenger)

else if (Associated(Stud%next)) then
   call Get_list_by_gender(Stud%next, List, ProcessedGenger)
else
   List => Null()
end if

!______Recursive_Selection_sort_by_Year_Of_Birth_and_Surname_____!
if (Associated(unsorted)) then
   call Choose_And_Paste(unsorted, unsorted, unsorted%next)
   call Sort_Group_List(unsorted%next)
end if

!______Choose_and_paste_at_the_head_the_maximum_element__________!
if (Associated(current)) then
   if (Swap(current, maximum)) then
      call Choose_And_Paste(unsorted, current, current%next)
   else
      call Choose_And_Paste(unsorted, maximum, current%next)
   end if
else
   if (.not. Associated(unsorted, maximum)) then
      tmp_student => maximum
      maximum => maximum%next
      tmp_student%next => unsorted
      unsorted => tmp_student
   end if
end if

!_________________Two_elemets_comparison_by_all_fields____________!
Swap = .false.
if (current%YOB < maximum%YOB) then
   Swap = .true.
else if (current%YOB == maximum%YOB) then
   if (current%Surname < maximum%Surname) then
      Swap = .true.
   else if (current%Surname==maximum%Surname .and. &
        current%Initials < maximum%Initials) then
      Swap = .true.
   end if
end if
end function Swap
```

2 Сравнение реализаций

По результатам выполнения задачи составлена сводная таблица, оценивающая **участки кода по обработке данных**, где:

- Сложность участка оценивалась как число строк кода.
- Производительность участка кода - обратное время выполнения этого участка кода.
- Эффективность участка кода - отношение производительности участка кода к его сложности.

При компиляции выбран уровень оптимизации **-O3**

Для решения данной задачи на современных микроархитектурах наиболее эффективно использовать **структуру массивов**, так как время обработки данных и эффективность участка кода по обработке данных у этой реализации - наилучшие.

Если планируется масшатибрование кода в коммерческих целях, то следует рассмотреть использование **массива структур**: сложность кода при этом минимальна, а производительность незначительно уступает более производительной реализации[20] - структуре массивов.

	Массив строк	Массив символов	**Массив структур**	Структура массивов	Массив структур с хвостовой рекурсией	Структура массивов с хвостовой рекурсией	Динамич. список
Сплошные данные	+	+	-	+	-	+	-
Регулярный доступ	+	+	-	+	-	+	-
Векторизация	+	+	+	+	+	+	-
Потенциальная векторизация	-	-	-	-	-	-	-
Время работы кода	341.92с	144.41с	74.36с	61.15с	572.33с	183.64с	161.11с
Сложность кода	67	88	40	53	45	61	66
Эффективность кода	43.7	78.7	336.2	308.5	38.8	89.3	94.0

Таблица 1: Сравнительная таблица реализаций структур данных

[20] на 18%

3 Заключение

Задачи, решённые в курсовой работе:

- отсортированные списки сформированы

- задача реализована с использованием структур данных, приведённых в **таблице 1**.

- проведён анализ на регулярный доступ к памяти

- проведён анализ на векторизацию

- проведён сравнительный анализ реализаций

В ходе выполнения работы была неверно выбран *"массив структур с хвостовой рекурсией"*. Для исправления ошибки была предпринята реализация с помощью *"структуры массивов с хвостовой рекурсией"*.

При разработке использовался процессор i5-1135G7, архитектура – x64 AMD64 (Intel 64), микроархитектура – Willow Cove, семейство микроархитектуры – Tiger Lake.

Выводы:

1. Цель по выбору структуры данных для формирования отсортированных списков студентов была достигнута, предпочтительная структура данных - **структура массивов**.

2. При разработке необходимо своевременно использовать метрики эффективности и производительности кода чтобы избежать ошибок и финансовых издержек.

3. Получен практический опыт использования различных структур данных.

4. Получен опыт по приведению кода к единому стилю оформления.