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»

Выполнила
студентка гр.в5130904/30030 В.С.Шестакова

Руководитель
старший преподаватель С.А.Федоров

«_____»_____ 20__ г.

Санкт-Петербург
2023

Оглавление

Задание (вариант 12)

Дан список группы в виде:

ФАМИЛИЯ	И. О.	ГОД РОЖДЕНИЯ	ПРОПИСКА	ПОЛ
15 симв.	5 симв.	4 симв.	1 симв.	1 симв.

Пример входного файла (в графе прописки буква П стоит у петербуржцев, С — у гостей Санкт-Петербурга):

Иванов	И. И.	1995	П	М
Петрова	Х. Л.	1994	С	Ж

Выделить из них трёх наиболее молодых петербуржцев мужчин. Пример выходного файла:

Иванов	И. И.	1995
Галкин	В. И.	1997
Потапкин	Е. З.	1997

Указание. Не следует сортировать весь список – нужно найти только первые три элемента. Например, можно три раза последовательно находить максимальный год рождения, но не учитывая уже найденные на предыдущем шаге (MaxLoc с mask).

В 1.6 необходимо сформировать три ссылки на самых молодых.

Введение

Цель работы — выбор структуры данных для решения поставленной зада- чи 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.

Глава 1.Реализация и анализ применения различных структур данных

Исходный список включал в себя список, в котором общее количество входных данных составило 480099.

1.Реализация задания с использованием массивов структур

При обработке данных осуществляется регулярный доступ к памяти, сами данные в памяти сплошные.

Объявление структуры данных

```
implicit none

integer, parameter                    :: AMOUNT = 84, SURNAME_LEN = 15, &
                                         INITIALS_LEN = 5, YEAR_Len = 4
character(kind=CH_), parameter        :: MALE = Char(1052, CH_), &
                                         SPb = Char (1055, CH_)
character(:), allocatable             :: input_file, output_file, format
character(SURNAME_LEN, kind=CH_)                    :: Surnames(AMOUNT) = ""
character(SURNAME_LEN, kind=CH_), allocatable       :: Boys_Surnames(:)
character(INITIALS_LEN, kind=CH_)                   :: Initials(AMOUNT) = ""
character(INITIALS_LEN, kind=CH_), allocatable      :: Boys_Initials(:)
character(kind=CH_)                                 :: Gender(AMOUNT) = ""
character(kind=CH_)                                 :: Regist(AMOUNT) = ""
integer                                             :: Years(AMOUNT) = 0, &
                                                       Boys_Amount = 0
integer, allocatable                  :: Boys_years(:), Boys_Pos(:), X(:)
logical, allocatable                  :: Boys_SPb(:), Mask_Boys(:)
integer                               :: In, Out, IO, i = 0, j = 0
integer, parameter                    :: INDEXES(*) = [(i, i = 1, AMOUNT)]
```

В конструкции DO CONCURRENT отдельные итерации цикла не имеют взаимозависимостей. Векторизация задействована на строке Mask_Boys(:) = .true.

Основные операторы обработки данных

```
Boys_SPb      = ((Gender == MALE) .and. (Regist == SPb))
Boys_Amount   = Count(Boys_SPb)
Boys_Pos      = Pack(INDEXES, Boys_SPb)

do concurrent (i = 1:Boys_Amount)
   Boys_Surnames(i)  = Surnames(Boys_Pos(i))
   Boys_Initials(i)  = Initials(Boys_Pos(i))
   Boys_years(i)  = Years(Boys_Pos(i))
end do

Mask_Boys(:) = .true.
do  j = 1, 3
    X(j) = MaxLoc(Boys_years(:),1,Mask_Boys(:))
    Mask_Boys(X(j)) = .false.
end do
```

2.Реализация задания с использованием массивов символов

Для обеспечения регулярного доступа к данным было выбрано назначение индексов Surnames(SURNAMES_LEN, AMOUNT), Initials(INITIALS_LEN, AMOUNT), т.к. будет вестись сравнение строк, матрица хранится по строкам – A(M, N). Тогда любая i-ая строка A(:, i) будет сплошной, обход массива выполняется по столбцам.

Объявление структуры данных

```
 implicit none

character(*), parameter           :: input_file = "../data/class.txt", &
                                     output_file = "output.txt"
integer, parameter                :: AMOUNT = 40336, Boys_F = 3, &
                                     SURNAMES_LEN = 15, INITIALS_LEN = 5
character(kind=CH_), parameter    :: MALE = Char(1052, CH_), &
                                      SPb = Char (1055, CH_)
character(kind=CH_), allocatable  :: Surnames(:, :), Initials(:, :), &
                                     Registr(:), Gender(:)
integer, allocatable              :: Year(:)
character(kind=CH_), allocatable  :: Boys_Surnames(:, :), &
                                     Boys_Initials(:, :)
integer, allocatable              :: Boys_Year(:)
```

Имеется зависимость чтение-после-записи.

Основные операторы обработки данных

```
Boys_SPb = (Gender == Gender_Sym .and. Registr == Regis_Sym)

do i = 1, BOYS_F
      Pos = MaxLoc(Year, 1, Boys_SPb)
      Boys_Surnames(:, i) = Surnames(:, Pos)
      Boys_Initials(:, i) = Initials(:, Pos)
      Boys_Year(i) = Year(Pos)
      Boys_SPb(Pos) = .false.
end do
```

3.Реализация задания с использованием массивов структур

Данные в памяти не являются сплошными.

Объявление структуры данных

```
implicit none
integer, parameter                          :: STUD_AMOUNT = 17425, &
            SURNAMES_LEN = 15, INITIALS_LEN = 5, BOYS_AMOUNT = 3
type student
        character(SURNAMES_LEN, kind=CH_)    :: Surnames                = ""
        character(INITIALS_LEN, kind=CH_)    :: Initials                = ""
        integer(I_)                          :: Year                    = 0
        character(kind=CH_)                  :: Registration            = ""
        character(kind=CH_)                  :: Gender                  = ""
end type student

character(kind=CH_), parameter              :: MALE = Char(1052, CH_), &
                                               SPb = Char(1055, CH_)
character(:), allocatable                   :: input_file, &
                                               output_file, data_file
type(student), allocatable                  :: Group(:)
type(student), allocatable                  :: Boys(:)
```

Основные операторы обработки данных

```
Boys_SPb = (Group%Registration == Regis_Sym .and. &
             Group%Gender == Gender_Sym)

 do i = 1, BOYS_AMOUNT
        Ind_Boy = MaxLoc(Group%Year, 1, Boys_SPb)
        Boys(i) = Group(Ind_Boy)
        Boys_SPb(Ind_Boy) = .false.
 end do
```

4.Реализация задания с использованием структур массивов

Данные в памяти являются сплошными.

Объявление структуры данных

```
implicit none
integer, parameter      :: STUD_AMOUNT = 40336, SURNAMES_LEN = 15, &
                           INITIALS_LEN = 5, BOYS_AMOUNT = 3
type student
       character(SURNAMES_LEN, kind=CH_),allocatable  :: Surnames(:)
       character(INITIALS_LEN, kind=CH_), allocatable :: Initials(:)
       integer(I_),allocatable                        :: Year(:)
       character(kind=CH_),allocatable                :: Registration(:)
       character(kind=CH_),allocatable                :: Gender(:)
  end type student

type boy
     character(SURNAMES_LEN, kind=CH_)    :: Surnames(BOYS_AMOUNT)  = ""
     character(INITIALS_LEN, kind=CH_)    :: Initials(BOYS_AMOUNT) = ""
     integer(I_)                          :: Year(BOYS_AMOUNT) = 0
end type boy

character(kind=CH_), parameter             :: MALE = Char(1052, CH_), &
                                              SPb = Char(1055, CH_)
character(:), allocatable                  :: input_file, output_file, &
                                              data_file
type(student)                              :: Group
type(boy)                                  :: Boys
```

Основные операторы обработки данных

```
Boys_SPb = (Group%Gender == Gender_Sym .and. &
            Group%Registration == Regis_Sym)

 do concurrent(i=1:BOYS_AMOUNT)
    Ind_Boy = MaxLoc(Group%Year, 1, Boys_Spb)
    Boys%Surnames(i) = Group%Surnames(Ind_Boy)
    Boys%Initials(i) = Group%Initials(Ind_Boy)
    Boys%Year(i) = Group%Year(Ind_Boy)

    Boys_SPb(Ind_Boy) = .false.
 end do
```

O3

5.Реализация задания с использованием структур массивов и с использованием хвостовой рекурсии при обработке данных

Для реализация выбрана структура массивов. Данные в памяти являются сплошными.

Объявление структуры данных

```
implicit none
integer, parameter          :: STUD_AMOUNT = 40336, SURNAMES_LEN = 15, &
                               INITIALS_LEN = 5, BOYS_AMOUNT = 3
  type student
       character(SURNAMES_LEN, kind=CH_),allocatable   :: Surnames(:)
       character(INITIALS_LEN, kind=CH_), allocatable  :: Initials(:)
       integer(I_),allocatable                         :: Year(:)
       character(kind=CH_),allocatable                 :: Registration(:)
       character(kind=CH_),allocatable                 :: Gender(:)
  end type student

type boy
  character(SURNAMES_LEN, kind=CH_)          :: Surnames(BOYS_AMOUNT)  = ""
  character(INITIALS_LEN, kind=CH_)          :: Initials(BOYS_AMOUNT) = ""
  integer(I_)                                :: Year(BOYS_AMOUNT) = 0
end type boy
character(kind=CH_), parameter               :: MALE = Char(1052, CH_), &
                                                SPb = Char(1055, CH_)
  type(student)                              :: Group
  type(boy)                                  :: Boys
  logical, allocatable                       :: Boys_SPb(:)
  integer                                    :: Count_mask
```

Основные операторы обработки данных

```
  Boys_SPb = ((Group%Gender == MALE) .and. (Group%Registration == SPb))

  Pos = MaxLoc(Group%Year, 1, Mask)

  Boys%Surnames(j) = Group%Surnames(Pos)
  Boys%Initials(j) = Group%Initials(Pos)
  Boys%Year(j) = Group%Year(Pos)

  Mask(Pos) = .false.

  if (j < Boys_Amount .and. Count(Mask) /=  0) &
      call Citizen(Group, Boys, Mask, j+1)
```

6.Реализация задания с с использованием однонаправленного динамического списка

Объявление структуры данных

```
implicit none

integer, parameter      :: STUD_AMOUNT = 5, SURNAMES_LEN = 15, &
                           INITIALS_LEN = 5, BOYS_AMOUNT = 3
type student
    character(SURNAMES_LEN, kind=CH_)    :: Surnames     = ""
    character(INITIALS_LEN, kind=CH_)    :: Initials     = ""
    integer(I_)                          :: Year         = 0
    character(kind=CH_)                  :: Registration = ""
    character(kind=CH_)                  :: Gender       = ""
    type (student), pointer              :: next         => Null()
end type student

 character(kind=CH_), parameter    :: MALE = Char(1052, CH_), &
                                      SPB = Char(1055, CH_)
 character(:), allocatable         :: input_file, output_file

 type(student), pointer            :: Group_List => Null()
 type(student), pointer            :: Boy_1 => Null()
 type(student), pointer            :: Boy_2 => Null()
 type(student), pointer            :: Boy_3 => Null()
```

Основные операторы обработки данных

```
if (Stud%Gender == Gender .and. &
      Stud%Registration == Regis) then
    if (.not.Associated(Boy_1).or.Stud%Year_Birth > &
         Boy_1%Year_Birth) then
             Boy_3 => Boy_2
             Boy_2 => Boy_1
             Boy_1 => Stud
     else if (.not.Associated(Boy_2).or.Stud%Year_Birth > &
               Boy_2%Year_Birth) then
                    Boy_3 => Boy_2
                    Boy_2 => Stud
  else if (.not. Associated(Boy_3) .or. Stud%Year_Birth > &
               Boy_3%Year_Birth) then
             Boy_3 => Stud
 end if
        if (Associated(Stud%next)) &
 call Get_Boys_Peter(Stud%next, Boy_1, Boy_2, Boy_3, Gender, Regis)
     else if (Associated(Stud%next)) then
 call Get_Boys_Peter(Stud%next, Boy_1, Boy_2, Boy_3, Gender, Regis)
   end if
```

Глава 2. Сравнение реализаций

По результатам выполнения задания создана сводная таблица реализаций по критериям: сплошные данные, регулярный доступ, векторизация, потенциальная векторизация, а также по показателям: время работы участка кода по обработке данных, сложность участка кода по обработке данных, эффективность участка кода по обработке данных.

Сложность участка кода равна количеству строк участка кода по обработке данных. Эффективность участка кода равна отношению производительности участка кода к сложности этого участка кода. Производительность участка кода равна обратному времени работы этого участка кода.

При реализации кода выбран уровень оптимизации -O3.

Структура данных	Сплошные данные	Регулярный доступ	Векторизация	Потенциальная векторизация	Время работы,с	Сложность	Эффективность
массив строк	+	+	+	+	4,916e-03	15	13
массив символов	+	+	-	+	1,773e-03	9	63
массив структур	-	-	-	+	7,226e-03	6	23
структура массивов	+	-	-	+	1,486e-03	8	84
структура массивов, хвостовая рекурсия	+	-	-	+	1,796e-03	8	70
динамический список	-	-	-	-	2,200e-05	16	2840

Таблица 1 - Результаты обработки данных

Выводы

В ходе выполнения работы получен список самых молодых юношей петербуржцев:

Иванов	Х. Т	2023
Лапиков	Н. В.	2023
Вергентьев	Т. В.	2022

При выполнении курсовой работы были решены следующие задачи:

1. Реализовано задание с использованием массивов символов;
2. Реализовано задание с использованием массивов структур;
3. Реализовано задание с использованием структур массивов;
4. Реализовано задание с использованием структур массивов и с использованием хвостовой рекурсии при обработке данных;
5. Реализовано задание с использованием однонаправленного динамического списка;
6. Проведен анализ на регулярный доступ к памяти;
7. Проведен анализ на векторизацию кода;
8. Проведен сравнительный анализ реализаций.

Наиболее предпочтительной структурой данных для реализации поставленной задачи на современных микроархитектурах является динамический список, т.к. показатели времени обработки и эффективности участка кода по обработке данных являются наиболее оптимальными. [1]

Таким образом, цель работы - выбор структуры данных для решения поставленной задачи на современных микроархитектурах - достигнута.

[1] модель процессора Ryzen 5 5600X, архитектура AMD64, микроархитектура Zen 3(Vermeer)