|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
| Практическое задание № 5 | | |
| по дисциплине «Структура данных и алгоритмы» | | |
| **Сортировка** | | |
|  | | |
|  | Бригада 1 | Быков богдан |
| Группа ПМ-22 | Брыкова Дарья |
| Вариант 3 | Чёрный антон |
| Преподаватели | Хиценко Валентина Павловна |
|
|  |
|  |
|  |
| Новосибирск, 2023 | | |

1. **Задание**

Упорядочить таблицу, построенную в практическом задании «Таблицы» (вариант 1.1в), по новому ключу – по не возрастанию частоты использования элемента методом быстрого поиска.

Используя раздел операторов, дополнить элементы таблицы числом раз использования каждого ключа. Для поиска элементов в таблице использовать последовательный поиск.

1. **Анализ задачи**
   1. **Входные данные задачи:**

A ∈ <текст на Си>

<текст на Си> = <заголовок> {<тело>}

<заголовок> = int main ()

<тело> = {<строка>}

<строка>=<определение переменной>|<оператор присваивания>|<условный оператор>

<определение переменной> = <тип> <имя> = A, A∈R; | <тип> <имя> [n], n∈R, n **⩽** 232

<тип> = int | bool | double | float

<имя> = {wi | ‘a’**⩽**wi **⩽**’z’, i **⩾** 1}

<оператор присваивания> = <имя> = <имя> {ci, i ∈ N **⋃** {0} }, ci=+<имя>;

<условный оператор> = if (<выражение> <операции сравнения> <выражение>

<выражение> = <имя> | A, A ∈ R

<операции сравнения> = < | <= | > | >= | ==

B {ai | ai  <элемент>, i **⩽** N}

<элемент> = <имя> <тип> <занимаемая память> <число компонент> <тип компонент> (<элемент> | пусто)  
<имя> = строка  
<тип> = строка  
<занимаемая память> = целое число  
<число компонент> = целое число

<тип компонент> = строка

* 1. **Выходные данные задачи:**

С {ai | ai  <элемент>}

<элемент> = <имя> <тип> <занимаемая память> <число компонент> <тип компонент> (<элемент> | пусто)  
<имя> = строка  
<тип> = строка  
<занимаемая память> = целое число  
<число компонент> = целое число

<тип компонент> = строка

<число вхождений> = целое число

* 1. **Математическое решение задачи:**

prev = пустая строка

При a = первая строка в A

Повторять

Если в строке а есть символ “=”, то

при prev = первое слово

повторять

если prev есть в B, то прибавить 1 к числу вхождений элемента prev

prev = следующее слово в A

prev = следующее слово в А

пока существует prev

иначе если в a есть “if(”, то

при prev = второе слово

повторять

если prev есть в B, то прибавить 1 к числу вхождений элемента prev

prev = следующее слово в A

prev = следующее слово в А

пока существует prev

a = следующее слово А

Пока существует а

Быстрая сортировка B по числу вхождение в порядке не возрастания

**Быстрая сортировка со стенками L и R**

**Piv(L,R,ri,rj)**

c =

при i = L и j = R

повторять

повторять i = i + 1 пока кол-во вхождений i-го элемента меньше кол-ва вхождений c

повторять j = j + 1 пока кол-во вхождений j-го элемента больше кол-ва вхождений c

если Ij

если число вхождений i-го и j-го элемента не равны, то

поменять местами i-й и j-й эл-ты

i = i + 1

j = j + 1

пока Ij

ri = i

rj = j

* 1. **Подзадачи**

1. **Структуры основных входных и выходных данных**

Внешнее представление

файл в котором представлена таблица t в данном формате:

t = {t i-х | t i-й = n <имя> <тип> MEM <кол-во> <сложный тип>”, где n**⩽**N – номер занимаемой элементом в таблице строки, <имя> - имя переменной, <тип> - тип переменной, mem∈N  - количество памяти, занимаемой переменной

<кол-во> = 0 | A, A ∈N

<сложный тип> = NULL | <тип>

<имя> = {wi | ‘a’**⩽**wi **⩽**’z’, i **⩾** 1}

<тип> = int | bool | double | float

}

также код программы на языке Си <код> представлен в файле code.txt в виде:

<код> = <заголовок> {<тело>}

<заголовок> = int main ()

<тело> = {<строка>}

<строка>=<объявление переменной>|<строка с присваиванием>|<условный оператор>|<иная строка>

<объявление переменной> = <тип> <имя> = A, A∈R; | <тип> <имя> [n], n∈R, n **⩽** 2^32

<тип> = int | bool | double | float

<имя> = {wi | ‘a’**⩽**wi **⩽**’z’, i **⩾** 1}

<строка с присваиванием> = <имя> = <имя> {ci, i ∈ N **⋃** {0} }, ci=+<имя>;

<условный оператор> = if (<имя> {ci, i ∈ N **⋃** {0} }, ci= <условие> <имя>)

<условие> = < | <= | > | >= | ==

<иная строка> = строка кода на языке Си, в которой нет оператора присваивания и условного оператора

Внутреннее представление входных данных:

таблица t представлена в файле table.txt в виде массива элементов elem размера N=19

struct elem {

char name;

char type;

int mem;

int k;

char type\_k;

int num = 0;}

1. **Алгоритм задачи**

**Алгоритм хеш-функции**

int h

len = strlen(a);

ans = 0;

for (I = 0; I < len; i++)

ans += (a[i]);

return (ans % N);

**Алгоритм добавление числа использования ключа в таблицу**

void add

k = H(a);

counter = 0;

sscanf(a, «%[^[{; ]», a);

while (strcmp(t[k%N].name, a) && counter!= N) {

k +=1;

counter += 1;

}

if (!strcmp(t[k % N].name, a)) {

t[k%N].num += 1;

}

**Алгоритм сортировка относительно опорного элемента**

void piv

c = t[(L+R)/2];

i = L;

j = R;

look += 1;

while (i <= j)

{

cmp += 1;

while (t[i].num > c.num) {

i++;

cmp += 1;

}

while (t[j].num < c.num) {

j--;

cmp += 1;

}

if (i <= j)

{

cmp++;

if (t[i].num != t[j].num) {

x = t[i];

t[i] = t[j];

t[j] = x;

beg += 3;

}

i++;

j--;

}

}

ri = i;

rj = j;

**Алгоритм быстрой сортировки**

void Quicksort

i = 0;

j = 0;

if (R – L > 0)

{

PIV(t, L, R, &I, &j);

if (L < j) Quicksort(t, L, j);

if (I < R) Quicksort(t, I, R);

1. **Структура программы**

Подпрограмма «Добавление числа использования ключа a в таблицу t»;

void add(elem\* t, char\* a);

Параметры: t – исходная таблица, a – ключ;

Возвращает: t – таблица.

Подпрограмма «Ввод таблицы t»;

void input(elem\* t);

Параметры: t – пустая таблица;

Возвращает: t – заполненная таблица;

Подпрограмма «Быстрая сортировка таблицы t с левой и правой стенками L и R»;

void Quicksort(elem\* t, int L, int R);

Параметры: t – исходная таблица, L – номер крайнего левого элемента таблицы t, R – номер крайнего элемента таблицы t;

Возвращает: t – таблица.

Подпрограмма «Вывод таблицы t»;

void vyvod(elem\* t);

Параметры: t – таблица;

Возвращает: t – таблица.

1. **Текст программы**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

const int N = 19;

FILE\* myfile;

int look = 0;

int cmp = 0;

int beg = 0;

struct elem {

char name[256];

char type[15];

int mem;

int k;

char type\_k[15];

int num = -1;

};

struct types {

char D[7];

int size;

};

int h(char\* a) {

int len = strlen(a);

int ans = 0;

for (int i = 0; i < len; i++)

ans += int(a[i]);

return (ans % N);

}

void vyvod(elem\* t) {

int i = 0;

myfile = fopen("output.txt", "a");

while (i < N) {

if (strcmp(t[i].name, ""))

fprintf(myfile, "%i %s %s %i %i %s %i\n", i, t[i].name, t[i].type, t[i].mem, t[i].k, t[i].type\_k, t[i].num);

i++;

}

fprintf(myfile, "%i %i %i\n\n", look, cmp, beg);

}

void add(elem\* t, char\* a) {

int k = h(a);

int counter = 0;

sscanf(a, "%[^[{; ]", a);

while (strcmp(t[k%N].name, a) && counter!= N) {

k +=1;

counter += 1;

}

if (!strcmp(t[k % N].name, a)) {

t[k%N].num += 1;

}

}

void input(elem\* t){

myfile = fopen("table.txt", "r");

int i = 0;

while (fscanf(myfile, "%i ", &i) != EOF) {

fscanf(myfile, "%s %s %i %i %s", t[i].name, t[i].type, &t[i].mem, &t[i].k,t[i].type\_k);

if (strcmp(t[i].name, "")) t[i].num = 0;

}

myfile = fopen("code.txt", "r");

char a[25];

char\* prev;

fgets(a, 25, myfile);

while (!feof(myfile)) {

if (strstr(a, "=")) {

prev = strtok(a, " ");

while (prev != NULL) {

add(t, prev);

prev = strtok(NULL, " ");

prev = strtok(NULL, " ;");

}

}

else if (strstr(a, "if(")) {

prev = strtok(a, " ");

prev = strtok(NULL," ){");

while (prev != NULL) {

add(t, prev);

prev = strtok(NULL, " ");

prev = strtok(NULL, " ){");

}

}

fgets(a, 25, myfile);

}

vyvod(t);

}

void piv(elem\* t, int L, int R, int\* ri, int\* rj)

{

elem c = t[(L+R)/2];

int i = L;

int j = R;

look += 1;

while (i <= j)

{

cmp += 1;

while (t[i].num > c.num) {

i++;

cmp += 1;

}

while (t[j].num < c.num) {

j--;

cmp += 1;

}

if (i <= j)

{

cmp++;

if (t[i].num != t[j].num) {

elem x = t[i];

t[i] = t[j];

t[j] = x;

beg += 3;

}

i++;

j--;

}

}

\*ri = i;

\*rj = j;

}

void Quicksort(elem\* t, int L, int R)

{

int i;

int j;

if (R - L > 0)

{

piv(t, L, R, &i, &j);

if (L < j) Quicksort(t, L, j);

if (i < R) Quicksort(t, i, R);

}

}

int main()

{

elem t[N];

for (int i = 0; i < N; i++) {

strcpy\_s(t[i].name, "");

}

input(t);

Quicksort(t, 0, N - 1);

vyvod(t);

}

1. **Откладка программы**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Входные данные (код) | Входные данные  (таблица) | Выходные данные | Назначение теста |
| 1 | int main()  {  int a1;   a1 = a1 + 1;  } | 13 a1 int 4 0 NONE | 0 a1 int 4 0 NONE 2 | Простейший тест |
| 2 | int main()  {  int a;  float b;  double c;  b = 1;  c = 5;  if( c > 2 )  {   pritnf(“Hello World”);  }  } | 2 a int 4 0 NONE  3 b float 4 0 NONE  4 c double 8 0 NONE | 0 c double 8 0 NONE 2  1 b float 4 0 NONE 1  2 a int 4 0 NONE 0 | Тест на работу программы с таблицей с не совпадающими частотами появления  переменных |
| 3 | int main()  {  int a[10];  float b;  a[1] = 1;  b = 3;  double c;  c = 4;  bool d;  d = 1;  } | 2 a array 4 10 int  3 b float 4 0 NONE  4 c double 8 0 NONE  5 d bool 1 0 NONE | 0 b float 4 0 NONE 1  1 c double 8 0 NONE 1  2 d bool 1 0 NONE 1  3 a array 4 10 int 1 | Тест на работу программы с таблицей с совпадающими частотами появления  переменных |
| 4 | int main()  {  int a1;  double b1;  float c1[12];  a1 = 1;  b1 = 1;  c1[1] = 1;  int a2;  double b2;  float c2;  a2 = 1;  b2 = a1 + 1;  c2 = 1;  c2 = c1[1] + b1 + a1;  int a3;  double b3;  float c3;  a3 = a1;  b3 = 1 + 1;  c3 = 1;  int a4;  double b4;  float c4;  a4 = 1;  b4 = 1;  c4 = 1;  int a5;  double b5;  float c5’  a5 = c3;  b5 = c4;  c5 = 1;  int a6;  double b6;  float c6;  a6 = 3;  b6 = 1;  c6 = c3;  int a7;  a7 = a1 + a1+ a1+ a1 + a1;  } | 0 a3 int 4 0 NONE  1 b3 double 8 0 NONE  2 c3 float 4 0 NONE  3 a4 int 4 0 NONE  4 b4 double 8 0 NONE  5 c4 float 4 0 NONE  6 a5 int 4 0 NONE  7 b5 double 8 0 NONE  8 c3 float 4 0 NONE  9 a6 int 4 0 NONE  10 b6 double 8 0 NONE  11 c6 float 4 0 NONE  12 a7 int 4 0 NONE  13 a1 int 4 0 NONE  14 b1 double 8 0 NONE  15 c1 array 4 12 float  16 a2 int 4 0 NONE  17 b2 double 8 0 NONE  18 c2 float 4 0 NONE | 0 a1 int 4 0 NONE 6  1 c3 float 4 0 NONE 3  2 c4 float 4 0 NONE 2  3 c1 array 4 12 float 2  4 c2 float 4 0 NONE 2  5 c6 float 4 0 NONE 1  6 a7 int 4 0 NONE 1  7 b2 double 8 0 NONE 1  8 a2 int 4 0 NONE 1  9 b1 double 8 0 NONE 1  10 b6 double 8 0 NONE 1  11 a4 int 4 0 NONE 1  12 b3 double 8 0 NONE 1  13 a3 int 4 0 NONE 1  14 b4 double 8 0 NONE 1  15 a6 int 4 0 NONE 1  16 b5 double 8 0 NONE 1  17 a5 int 4 0 NONE 1 | Тест на работу программы с заполненной таблицей причем есть переменные появляющиеся различное количество раз |
| 5 | int main()  {  int a1;  double b1;  float c1;  a1 = 1;  b1 = 1;  c1 = 1;  int a2;  double b2;  float c2;  a2 = 1;  b2 = 1;  c2 = 1;  int a3;  double b3;  float c3;  a3 = 1;  b3 = 1;  c3 = 1;  int a4;  double b4;  float c4;  a4 = 1;  b4 = 1;  c4 = 4;  int a5;  double b5;  float c5;  a5 = 1;  b5 = 12;  int a6;  double b6;  float c6;  a6 = 1;  b6 = 1;  c6 = 1;  int a7;  a7 = 15;  if( a7 < 5) {  c6 = a7 + 15;  a5 = b5 + a5;  }  b5 = b5 – a5;  a5 = 0;  a5 = 121;  a5 = a7;  c6 = 0;  c6 = 12131;  if( b5 != 0 ) {  a6 = 0;  a1 = 1;  a2 = -1;  a2 = a1 + 1;  }  } | 0 a1 int 4 0 NONE  1 b1 double 8 0 NONE  2 c1 float 4 0 NONE  3 a2 int 4 0 NONE  4 b2 double 8 0 NONE  5 c2 float 4 0 NONE  6 a3 int 4 0 NONE  7 b3 double 8 0 NONE  8 c3 float 4 0 NONE  9 a4 int 4 0 NONE  10 b4 double 8 0 NONE  11 c4 float 4 0 NONE  12 a5 int 4 0 NONE  13 a1 int 4 0 NONE  14 b1 double 8 0 NONE  15 c1 array 4 12 float  16 a2 int 4 0 NONE  17 b2 double 8 0 NONE  18 c2 float 4 0 NONE | 0 a5 int 4 0 NONE 7  1 a2 int 4 0 NONE 4  2 a1 int 4 0 NONE 4  3 b1 double 8 0 NONE 2  4 c4 float 4 0 NONE 1  5 b4 double 8 0 NONE 1  6 c2 float 4 0 NONE 1  7 c1 array 4 12 float 1  8 b2 double 8 0 NONE 1  9 a4 int 4 0 NONE 1  10 a3 int 4 0 NONE 1  11 c1 float 4 0 NONE 1  12 c3 float 4 0 NONE 1  13 b3 double 8 0 NONE 1  14 b1 double 8 0 NONE 0  15 a1 int 4 0 NONE 0  16 c2 float 4 0 NONE 0  17 b2 double 8 0 NONE 0  18 a2 int 4 0 NONE 0 | Тест на работу программы с заполненной таблицей. |
| 6 | int main()  {  int a1;  int a2;  int a3;  int a4;  int a5;  int a6;  int a7;  int a8;  int a9;  int a10;  int a11;  int a12;  int a13;  int a14;  int a15;  int a16;  int a17;  int a18;  int a19;  } | 0 a7 int 4 0 NONE  1 a8 int 4 0 NONE  2 a9 int 4 0 NONE  3 a19 int 4 0 NONE  4 a10 int 4 0 NONE  5 a11 int 4 0 NONE  6 a12 int 4 0 NONE  7 a13 int 4 0 NONE  8 a14 int 4 0 NONE  9 a15 int 4 0 NONE  10 a16 int 4 0 NONE  11 a17 int 4 0 NONE  12 a18 int 4 0 NONE  13 a1 int 4 0 NONE  14 a2 int 4 0 NONE  15 a3 int 4 0 NONE  16 a4 int 4 0 NONE  17 a5 int 4 0 NONE  18 a6 int 4 0 NONE |  | Таблица упорядочена на входе |
| 7 | int main()  {  int a1;  int a2;  int a3;  int a4;  int a5;  int a6;  int a7;  int a8;  int a9;  int a10;  int a11;  int a12;  int a13;  int a14;  int a15;  int a16;  int a17;  int a18;  int a19;  a7 = 1;  a8 = 2;  a8 = 2;  a9 = 3;  a9 = 3;  a9 = 3;  a19 = 4;  a19 = 4;  a19 = 4;  a19 = 4;  a10 = 5;  a10 = 5;  a10 = 5;  a10 = 5;  a10 = 5;  a11 = 6;  a11 = 6;  a11 = 6;  a11 = 6;  a11 = 6;  a11 = 6;  a12 = 7;  a12 = 7;  a12 = 7;  a12 = 7;  a12 = 7;  a12 = 7;  a12 = 7;  a13 = 8;  a13 = 8;  a13 = 8;  a13 = 8;  a13 = 8;  a13 = 8;  a13 = 8;  a13 = 8;  a14 = 9;  a14 = 9;  a14 = 9;  a14 = 9;  a14 = 9;  a14 = 9;  a14 = 9;  a14 = 9;  a14 = 9;  a15 = 10;  a15 = 10;  a15 = 10;  a15 = 10;  a15 = 10;  a15 = 10;  a15 = 10;  a15 = 10;  a15 = 10;  a15 = 10;  a16 = 11;  a16 = 11;  a16 = 11;  a16 = 11;  a16 = 11;  a16 = 11;  a16 = 11;  a16 = 11;  a16 = 11;  a16 = 11;  a16 = 11;  a17 = 12;  a17 = 12;  a17 = 12;  a17 = 12;  a17 = 12;  a17 = 12;  a17 = 12;  a17 = 12;  a17 = 12;  a17 = 12;  a17 = 12;  a17 = 12;  a18 = 13;  a18 = 13;  a18 = 13;  a18 = 13;  a18 = 13;  a18 = 13;  a18 = 13;  a18 = 13;  a18 = 13;  a18 = 13;  a18 = 13;  a18 = 13;  a18 = 13;  a1 = 14;  a1 = 14;  a1 = 14;  a1 = 14;  a1 = 14;  a1 = 14;  a1 = 14;  a1 = 14;  a1 = 14;  a1 = 14;  a1 = 14;  a1 = 14;  a1 = 14;  a1 = 14;  a2 = 15;  a2 = 15;  a2 = 15;  a2 = 15;  a2 = 15;  a2 = 15;  a2 = 15;  a2 = 15;  a2 = 15;  a2 = 15;  a2 = 15;  a2 = 15;  a2 = 15;  a2 = 15;  a2 = 15;  a3 = 16;  a3 = 16;  a3 = 16;  a3 = 16;  a3 = 16;  a3 = 16;  a3 = 16;  a3 = 16;  a3 = 16;  a3 = 16;  a3 = 16;  a3 = 16;  a3 = 16;  a3 = 16;  a3 = 16;  a3 = 16;  a4 = 17;  a4 = 17;  a4 = 17;  a4 = 17;  a4 = 17;  a4 = 17;  a4 = 17;  a4 = 17;  a4 = 17;  a4 = 17;  a4 = 17;  a4 = 17;  a4 = 17;  a4 = 17;  a4 = 17;  a4 = 17;  a4 = 17;  a5 = 18;  a5 = 18;  a5 = 18;  a5 = 18;  a5 = 18;  a5 = 18;  a5 = 18;  a5 = 18;  a5 = 18;  a5 = 18;  a5 = 18;  a5 = 18;  a5 = 18;  a5 = 18;  a5 = 18;  a5 = 18;  a5 = 18;  a5 = 18;  a6 = 19;  a6 = 19;  a6 = 19;  a6 = 19;  a6 = 19;  a6 = 19;  a6 = 19;  a6 = 19;  a6 = 19;  a6 = 19;  a6 = 19;  a6 = 19;  a6 = 19;  a6 = 19;  a6 = 19;  a6 = 19;  a6 = 19;  a6 = 19;  a6 = 19;  } | 0 a7 int 4 0 NONE  1 a8 int 4 0 NONE  2 a9 int 4 0 NONE  3 a19 int 4 0 NONE  4 a10 int 4 0 NONE  5 a11 int 4 0 NONE  6 a12 int 4 0 NONE  7 a13 int 4 0 NONE  8 a14 int 4 0 NONE  9 a15 int 4 0 NONE  10 a16 int 4 0 NONE  11 a17 int 4 0 NONE  12 a18 int 4 0 NONE  13 a1 int 4 0 NONE  14 a2 int 4 0 NONE  15 a3 int 4 0 NONE  16 a4 int 4 0 NONE  17 a5 int 4 0 NONE  18 a6 int 4 0 NONE | 0 a6 int 4 0 NONE 19  1 a5 int 4 0 NONE 18  2 a4 int 4 0 NONE 17  3 a3 int 4 0 NONE 16  4 a2 int 4 0 NONE 15  5 a1 int 4 0 NONE 14  6 a18 int 4 0 NONE 13  7 a17 int 4 0 NONE 12  8 a16 int 4 0 NONE 11  9 a15 int 4 0 NONE 10  10 a14 int 4 0 NONE 9  11 a13 int 4 0 NONE 8  12 a12 int 4 0 NONE 7  13 a11 int 4 0 NONE 6  14 a10 int 4 0 NONE 5  15 a9 int 4 0 NONE 3  16 a8 int 4 0 NONE 2  17 a19 int 4 0 NONE 2  18 a7 int 4 0 NONE 1  12 32 63 |  |