Міністерство освіти і науки, молоді та спорту України

Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №1**

З дисципліни «СП»

Виконав: Перевірив:

Студент групи ІО-21 доц. Пустоваров В.І.

Коноз А.О.

Дата здачі\_\_\_\_\_\_\_\_\_\_\_\_\_

Захищено з балом\_\_\_\_\_

Київ 2014

**Завдання**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 12 | unsigned int | char\*\_ unsigned char | double | Черговий | |
| 2 | Спiвпадіння максимальної кількості літер без урахування їх послідовності у ключі. | | | |

**Лістинг коду**

#include "stdafx.h"

#include "tables.h"

#include "vistab.h"

// кількість елементів в масиві

int nTbStud = 2;

// розмір масиву

int lnTbStud = 100;

struct recrd tbStud[100] = {{{"Abrahams",82}, 5.0, 0}, {{"Dickens",84}, 6.5, 0}};

struct recrd tstArg = {{"Morgan",81}, 3.0, 0},

insArg = {{"Smith",89}, 5.2, 0},

insArg0 = {{"Clark", 54}, 4.5, 0},

insArg1 = {{"Stivenson", 8}, 3.1, 0},

insArg2={{"Doom",81},3.2,0},

insArg3={{"hamsAbra",85}, 4.3, 0};

int main(int argc, char\* argv[])

{

prRow(&tstArg);

prRow(selNmb(tbStud, 1));

prRow(selLin(tstArg.key, tbStud, 2));

prRow(insNmb(&insArg, tbStud, 3, &nTbStud));

prTab(tbStud, 0, nTbStud - 1);

prRow(updNmb(&insArg1, tbStud, 2, &nTbStud));

prRow(delNmb(tbStud, 1));

prTab(tbStud, 0, nTbStud - 1);

prRow(insLin(&insArg2, tbStud, &nTbStud));

prTab(tbStud, 0, nTbStud - 1);

prRow(delLin(&insArg1, tbStud, &nTbStud));

prTab(tbStud, 0, nTbStud - 1);

prRow(updLin(insArg.key, &insArg0, tbStud, &nTbStud));

prTab(tbStud, 0, nTbStud - 1);

prRow(pckLin(tbStud, &nTbStud));

prTab(tbStud, 0, nTbStud - 1);

prRow(srtBin(tbStud, nTbStud));

prTab(tbStud, 0, nTbStud - 1);

prRow(selBin(insArg2.key, tbStud, nTbStud));

prTab(tbStud, 0, nTbStud - 1);

prRow(selBinSame(insArg3.key, tbStud, nTbStud));

prClr(blue);

return 0;

}

#include "stdafx.h"

#include "tables.h"

#include <string.h>

struct recrd emptyElm = {"", 0, 0.0};

//------------------------------------------------------

// обробка записів таблиць за прямою адресою

// вибірка за прямою адресою

struct recrd\* selNmb(struct recrd\*tb, unsigned int nElm)

{

return &tb[nElm];

}

// включення за прямою адресою

struct recrd\* insNmb(struct recrd\*pElm,

struct recrd\*tb, unsigned int nElm, int \*pQnElm)

{

unsigned long n=\*pQnElm;

if (nElm < n) return 0;

while (nElm < n) tb[nElm++] = emptyElm;

tb[nElm++]=\*pElm;

\*pQnElm = nElm;

return &tb[nElm - 1];

}

// вилучення за прямою адресою

struct recrd\* delNmb(struct recrd\*tb, unsigned int nElm)

{

tb[nElm].\_del = -1;

return &tb[nElm];

}

// корекція за прямою адресою

struct recrd\* updNmb(struct recrd\*pElm,

struct recrd\*tb, unsigned int nElm, int \*pQnElm)

{

unsigned long n=\*pQnElm;

if (nElm >= n) return 0;

tb[nElm]=\*pElm;

return &tb[nElm];

}

//------------------------------------------------------

// порівняння за відношенням порядку

int neqKey(struct recrd\* el, struct keyStr kArg)

{

return (strcmp(el -> key.str, kArg.str) || el -> key.nMod != kArg.nMod);

}

// вибірка за лінійним пошуком

struct recrd\* selLin(struct keyStr kArg,

struct recrd\* tb, int ln)

{

int count = -1;

//цикл зупиниться, коли знайде еквівалентний ключ

while(++count < ln&&neqKey(&tb[count], kArg));

if (count >= ln) return 0;

return &tb[count];

}

// включення за лінійним пошуком

struct recrd\*insLin(struct recrd\*pElm,

struct recrd\*tb, int\*pQnElm)

{

int n = 0;

while (n<\*pQnElm&&tb[n].key.str[0] != 0) n++;

if (n==\*pQnElm) (\*pQnElm)++;

tb[n]=\*pElm;

return &tb[n];

}

// вилучення за лінійним пошуком

struct recrd\*delLin(struct recrd\*pElm,

struct recrd\*tb, int\*pQnElm)

{

struct recrd\*pEl = selLin(pElm -> key, tb, \*pQnElm);

if (pEl) pEl -> \_del = -1;

return pEl;

}

// корекція за лінійним пошуком

struct recrd\*updLin(struct keyStr kArg, struct recrd\*pElm,

struct recrd\*tb, int\*pQnElm)

{

struct recrd\*pEl = selLin(kArg, tb, \*pQnElm);

if (pEl)\*pEl=\*pElm;

return pEl;

}

// пакування для лінійного пошуку

struct recrd\*pckLin(struct recrd\*tb, int\*pQnElm)

{

int n = 0, ln = 0;

while (n<\*pQnElm)

{

if (tb[n].\_del == 0)

{

tb[ln] = tb[n];

ln++;

}

n++;

}

\*pQnElm=ln;

return tb;

}

//------------------------------------------------------

// порівняння рядків

int cmpStr(unsigned char\* s1, unsigned char\* s2)

{

unsigned n = 0;

while (s1[n] == s2[n] && s1[n] != 0) n++;

return s1[n] - s2[n];

}

// порівняння за відношенням порядку рядків

int cmpKey(struct recrd\* el, struct keyStr kArg)

{

int i = cmpStr((unsigned char\*)el -> key.str, (unsigned char\*)kArg.str);

if (i) return i;

return el -> key.nMod - kArg.nMod;

}

// порівняння за відношенням порядку

int cmpKys(struct keyStr \*k0, struct keyStr \*kArg)

{

int i = cmpStr((unsigned char\*)k0 -> str,

(unsigned char\*)kArg -> str);

if (i) return i;

return k0 -> nMod - kArg -> nMod;

}

//------------------------------------------------------

// сортування для двійкового пошуку

struct recrd\*srtBin(struct recrd\*tb, int ln)

{

int n = 0, n1;

struct recrd el;

for (; n < ln; n++)

for (n1 = n + 1; n1 < ln; n1++)

if (cmpKey(&tb[n], tb[n1].key) > 0)

{

el = tb[n];

tb[n] = tb[n1];

tb[n1] = el;

}

return tb;

}

// вибірка за двійковим пошуком

struct recrd\*selBin(struct keyStr kArg, struct recrd\*tb, int ln)

{

int i, nD = -1, nU = ln, n = nD + nU>>1;

while (i = cmpKey(&tb[n], kArg))

{

if (i > 0) nU=n;

else nD = n;

n = (nD + nU) >> 1;

if (n == nD) return NULL;

}

return &tb[n];

}

// включення за двійковим пошуком

// При відсутності елементу звільнити для нього місце,

// просунувши останні елементи в таблиці на один запис вперед

// Записати новий елемент на звільнене місце

struct recrd\*insBin(struct recrd\*pElm,

struct recrd\*tb, int\*pQnElm)

{

struct recrd\*pr = selBin(pElm -> key, tb, \*pQnElm);

if (pr)

{

struct fStr s=pElm -> func;

pr -> func = s;

return pr;

}

int n;

if (pr == NULL)

for (n=\*pQnElm; n > 0; n--)

tb[n] = tb[n - 1];

tb[n]=\*pElm;

return &tb[n];

}

// вилучення за двійковим пошуком

// при знаходженні елементу помічається для вилучення

struct recrd\*delBin(struct recrd\*pElm,

struct recrd\*tb, int\*pQnElm)

{

struct recrd\*pr = selBin(pElm -> key, tb, \*pQnElm);

if (pr) pr -> \_del = -1;

return pr;

}

// корекція за двійковим пошуком

// при знаходженні елементу заповнюється новими значеннями полів

struct recrd\*updBin(struct keyStr, struct recrd\*pElm,struct recrd\*tb, int\*pQnElm)

{

struct recrd\*pr = selBin(pElm -> key, tb, \*pQnElm);

if (pr)\*pr=\*pr;

return pr;

}

int compareKeys(unsigned char\* s1, unsigned char\* s2){

// unsigned char s2copy = \*s2;

int s1Size = 0;

int count = 0;

for(int i = 0; s1[i] != 0; i++){

s1Size++;

for(int j = 0; s2[j] != 0; j++){

if(s1[i] == s2[j]){

count++;

// s2[j] = '-';

break;

}

}

}

return count;

}

int countMaxSame(struct keyStr \*k0, struct keyStr \*kArg)

{

int i = compareKeys((unsigned char\*)k0 -> str,

(unsigned char\*)kArg -> str);

return i;

}

struct recrd\*selBinSame(struct keyStr kArg, struct recrd\*tb, int ln)

{

int same = 0;

int position = 0;

int temp = 0;

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

temp = countMaxSame(&kArg, &tb[i].key);

if(temp > same){

same = temp;

position = i;

}

}

return &tb[position];

}

#include "stdafx.h"

#include "vistab.h"

#include "tables.h"

#include <stdio.h>

void prClr(enum color cl)

{

char\* sClr[6] =

{

"violet","blue","green","yellow","orange","red"

};

printf("%s", sClr[cl]);

}

void prRow(struct recrd\* rw)

{

if (rw == 0) printf("is absent\n");

else

if (rw -> \_del) printf("is deleted\n");

else printf("%-10s %3u %5.3f\n", rw -> key.str, rw -> key.nMod, rw -> func.\_f);

}

void prTab(struct recrd\* rw, int n0, int n1)

{

printf(" Table:\n");

while (n0 <= n1)

{

prRow(rw + n0);

n0++;

}

}

struct keyStr // ключова частина запису

{

char\* str; // ключові поля

int nMod;

};

struct fStr // функціональна частина запису

{

float \_f;

};

struct recrd // структура рядка таблиці

{

struct keyStr key; // примірник структури ключа

struct fStr func; // примірник функціональної частини

char \_del; // ознака вилучення

};

//------------------------------------------------------

// обробка записів таблиць за прямою адресою

// вибірка за прямою адресою

struct recrd\* selNmb(struct recrd\*, unsigned int nElm);

// включення за прямою адресою

struct recrd\* insNmb(struct recrd\*pElm,

struct recrd\*tb, unsigned int nElm, int\*pQnElm);

// вилучення за прямою адресою

struct recrd\* delNmb(struct recrd\*, unsigned int nElm);

// корекція за прямою адресою

struct recrd\* updNmb(struct recrd\*pElm,

struct recrd\*tb, unsigned int nElm, int\*pQnElm);

//------------------------------------------------------

// порівняння за відношенням нерівності

int neqKey(struct recrd\*, struct keyStr);

// порівняння рядків

int cmpStr(unsigned char\* s1, unsigned char\* s2);

// порівняння за відношенням порядку

int cmpKey(struct recrd\*, struct keyStr);

// порівняння за відношенням порядку

int cmpKys(struct keyStr\*, struct keyStr\*);

// порівняння за відношенням схожості

int simKey(struct recrd\*, struct keyStr);

//------------------------------------------------------

// вибірка за лінійним пошуком

struct recrd\*selLin(struct keyStr, struct recrd\*tb, int ln);

// включення за лінійним пошуком

struct recrd\*insLin(struct recrd\*pElm,

struct recrd\*tb, int\*pQnElm);

// вилучення за лінійним пошуком

struct recrd\*delLin(struct recrd\*pElm,

struct recrd\*tb, int\*pQnElm);

// корекція за лінійним пошуком

struct recrd\*updLin(struct keyStr, struct recrd\*pElm,

struct recrd\*tb, int\*pQnElm);

// пакування для лінійного пошуку

struct recrd\*pckLin(struct recrd\*tb, int\*pQnElm);

//------------------------------------------------------

// сортування за двійковим пошуком

struct recrd\*srtBin(struct recrd\*, int ln);

// вибірка за двійковим пошуком

struct recrd\*selBin(struct keyStr, struct recrd\*, int ln);

// включення за двійковим пошуком

struct recrd\*insBin(struct recrd\*pElm,

struct recrd\*tb, int\*pQnElm);

// вилучення за двійковим пошуком

struct recrd\*delBin(struct recrd\*pElm,

struct recrd\*tb, int\*pQnElm);

// корекція за двійковим пошуком

struct recrd\*updBin(struct keyStr, struct recrd\*pElm,

struct recrd\*tb, int\*pQnElm);

struct recrd\*selBinSame(struct keyStr kArg, struct recrd\*tb, int ln);

-