|  |
| --- |
| Київський національний університет  «Київський політехнічний інститут» |
| Факультет інформатики та обчислювальної техніки |
| Кафедра обчислювальної техніки |
| Лабораторна робота №1 |
| з дисципліни «Системне програмування» |
| *Залікова книжка № 0103* |
|  |
| Виконав студент 3-го курсу |
| групи ІО-01 |
| Редько Олександр |

|  |
| --- |
| 2012 |

Мета: Вивчення типів таблиць (для пошуку за прямою адресою та лінійного і двійкового пошуку) в системних програмах і конструкцій базової мови програмування для їх визначення та обробки відповідними методами.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № вар. | Тип ключа для прямої адреси | Тип ключа для інших видів пошуку | Тип функціо­нального поля | Тип вибірки |
| 3 | unsigned char | char\*\_ unsigned long | Struct | Черговий |

# Лістинг програми

## Sysprog\_Lab1.cpp

// Sysprog\_Lab1.cpp : Defines the entry point for the console application.

#include "stdafx.h"

#include "tables.h"

#include "vistab.h"

/\*\*

\* System programming. Laboratory work #1

\*

\* author Redko Alexander

\* group IO-01

\* date 06.10.12

\*

\*/

int nTbStud = 5, lnTbStud = 100;

struct record tbStud[100] =

{

{{"Diatlov", 89}, {4}, 0},

{{"Kutsar", 89}, {12}, 0},

{{"Redko", 56}, {7}, 0},

{{"Chychno", 12}, {9}, 0},

{{"Melnyk", 32}, {4}, 0}

};

struct record tstArg=

{{"Melnyk", 32}, {2}, 0},

insArg = {{"Gazizov", 89}, {3}, 0},

insLin = {{"Dudko", 80}, {4}, 0},

delLin = {{"Chychno", 12}, {3}, 0},

insArg1 = {{"Bilyk", 8}, {4}, 0},

updLin = {{"Sokol", 5}, {12}, 0},

selBin = {{"Kutsar", 89}, {8}, 0},

insBin = {{"Naumova", 1}, {1}, 0},

updBin = {{"Karas'", 8}, {9}, 0},

sel2Bin = {{"Melnyk", 32}, {4}, 0},

SelLab1={{"Ed", 2}, {41}, 0};

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

{

printTable(tbStud, 0, nTbStud - 1);

printf("-------------------------------------------------------------------");

printf("\nSelect by direct address element #1: ");

printRow(selectNumber(tbStud, 1));

printf("\nInsert row by direct address: ");

printRow(insertNumber(&insArg, tbStud, 5, &nTbStud));

printTable(tbStud, 0, nTbStud - 1);

printf("\nUpdating row by direct address #2: ");

printRow(updateNumber(&insArg1,tbStud, 2, &nTbStud));

printTable(tbStud, 0, nTbStud - 1);

printf("\nDelete row by direct address #2: \n");

deleteNumber(tbStud, 2);

printTable(tbStud, 0, nTbStud - 1);

printf("-------------------------------------------------------------------");

printf("\nInsert linear: ");

printRow(insertLinear(&insLin, tbStud, &nTbStud));

printTable(tbStud, 0, nTbStud - 1);

printf("\nSelect linear by key: ");

printRow(selectLinear(tstArg.key, tbStud, 5));

printf("\nDelete linear by key: ");

printRow(deleteLinear(&delLin, tbStud, &nTbStud));

printTable(tbStud, 0, nTbStud - 1);

printf("\nUpdate linear by key: ");

printRow(updateLinear(insArg.key, &updLin, tbStud, &nTbStud));

printTable(tbStud, 0, nTbStud - 1);

printf("\nAfter linear pack: \n");

packLinear(tbStud, &nTbStud);

printTable(tbStud, 0, nTbStud - 1);

printf("-------------------------------------------------------------------");

printf("\nBinary sort by keys: \n");

sortBinary(tbStud, nTbStud);

printTable(tbStud, 0, nTbStud - 1);

printf("\nSelect by binary search: ");

printRow(selectBinary(selBin.key, tbStud, nTbStud));

printTable(tbStud, 0, nTbStud - 1);

printf("\nInserting by binary search: ");

printRow(insertBinary(&insBin, tbStud, &nTbStud));

printTable(tbStud, 0, nTbStud);

printf("\nDelete by binary search: ");

printRow(deleteBinary(&selBin, tbStud, &nTbStud));

printTable(tbStud, 0, nTbStud);

printf("\nUpdate by binary search: ");

printRow(updateBinary(sel2Bin.key, &updBin, tbStud, &nTbStud));

printTable(tbStud, 0, nTbStud);

return 0;

}

## tables.h

// ключова частина запису

struct keyString

{

char \*str;

unsigned long nMod;

};

// функціональна частина запису

struct functionString

{

struct

{

int i;

} \_f ;

};

// структура рядка таблиці

struct record

{

// примірник структури ключа

struct keyString key;

// примірник функціональної частини

struct functionString func;

// ознака вилучення

char \_del;

};

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

// обробка записів таблиць за прямою адресою

// вибірка за прямою адресою

struct record \*selectNumber(struct record \*, unsigned char);

// включення за прямою адресою

struct record \*insertNumber(struct record \*, struct record \*, unsigned char, int \*);

// вилучення за прямою адресою

struct record \*deleteNumber(struct record \*, unsigned char);

// корекція за прямою адресою

struct record \*updateNumber(struct record \*, struct record \*, unsigned char, int \*);

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

// порівняння за відношенням нерівності

int neqKey(struct record \*, struct keyString);

// порівняння рядків

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

// порівняння за відношенням порядку

int cmpKey(struct record \*, struct keyString);

// порівняння за відношенням порядку

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

// порівняння за відношенням схожості

int simKey(struct record \*, struct keyString);

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

// вибірка за лінійним пошуком

struct record \*selectLinear(struct keyString, struct record \*, int);

// включення за лінійним пошуком

struct record \*insertLinear(struct record \*, struct record \*, int \*);

// вилучення за лінійним пошуком

struct record \*deleteLinear(struct record \*, struct record \*, int \*);

// корекція за лінійним пошуком

struct record \*updateLinear(struct keyString, struct record \*, struct record \*, int \*);

// пакування для лінійного пошуку

struct record \*packLinear(struct record \*, int \*);

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

// сортування за двійковим пошуком

struct record \*sortBinary(struct record \*, int);

// вибірка за двійковим пошуком

struct record \*selectBinary(struct keyString, struct record \*, int);

// включення за двійковим пошуком

struct record \*insertBinary(struct record \*, struct record \*, int \*);

// вилучення за двійковим пошуком

struct record \*deleteBinary(struct record \*, struct record \*, int \*);

// корекція за двійковим пошуком

struct record \*updateBinary(struct keyString, struct record \*, struct record \*, int \*);

## tables.cpp

#include "stdafx.h"

#include "tables.h"

#include <string.h>

struct record emptyElm = {{"", 0}, {0}};

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

// обробка записів таблиць за прямою адресою

// вибірка за прямою адресою

struct record\* selectNumber(struct record \*tb, unsigned char nElm)

{

return &tb[nElm];

}

// включення за прямою адресою

struct record\* insertNumber(struct record \*pElm, struct record \*tb, unsigned char nElm, int \*pQnElm)

{

int 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 record\* deleteNumber(struct record \*tb, unsigned char nElm)

{

tb[nElm].\_del = -1;

return &tb[nElm];

}

// корекція за прямою адресою

struct record \*updateNumber(struct record \*pElm, struct record \*tb, unsigned char nElm, int \*pQnElm)

{

int n = \*pQnElm;

if(nElm >= n)

return 0;

tb[nElm] = \*pElm;

return &tb[nElm];

}

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

// порівняння за відношенням порядку

int neqKey(struct record \*el, struct keyString kArg)

{

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

}

// вибірка за лінійним пошуком

struct record\* selectLinear(struct keyString kArg, struct record \*tb, int ln)

{

while(--ln >= 0 && neqKey(&tb[ln], kArg));

if(ln < 0)

return 0;

return &tb[ln];

}

// включення за лінійним пошуком

struct record \*insertLinear(struct record \*pElm, struct record \*tb, int \*pQnElm)

{

unsigned long n = 0;

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

n++;

if(n == \*pQnElm)

(\*pQnElm)++;

tb[n] = \*pElm;

return &tb[n];

}

// вилучення за лінійним пошуком

struct record \*deleteLinear(struct record \*pElm, struct record \*tb, int \*pQnElm)

{

struct record \*pEl = selectLinear(pElm->key, tb, \*pQnElm);

if(pEl)

pEl->\_del=-1;

return pEl;

}

// корекція за лінійним пошуком

struct record \*updateLinear(struct keyString kArg, struct record \*pElm, struct record \*tb, int \*pQnElm)

{

struct record \*pEl = selectLinear(kArg, tb, \*pQnElm);

if(pEl)

\*pEl = \*pElm;

return pEl;

}

// пакування для лінійного пошуку

struct record \*packLinear(struct record \*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 рядкыв

int cmpKey(struct record\* el, struct keyString 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 keyString \*k0, struct keyString \*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 record \*sortBinary(struct record \*tb, int ln)

{

int n = 0, n1;

struct record 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 record \*selectBinary(struct keyString kArg, struct record \*tb, int ln)

{

int i, nD = -1, nU = ln, n = nD + nU/2;

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

struct record \*insertBinary(struct record \*pElm, struct record \*tb, int \*pQnElm)

{

struct record \*pr = selectBinary(pElm->key, tb, \*pQnElm);

if(pr)

{

struct functionString s = pElm->func;

pr->func = s;

return pr;

}

// При відсутності елементу звільнити для нього місце,

// просунувши останні елементи в таблиці на одне заипс вперед

// Записати новий елемент на звільнене місце

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

struct record \*deleteBinary(struct record \*pElm, struct record\*tb, int\*pQnElm)

{

struct record \*pr = selectBinary(pElm->key, tb, \*pQnElm);

// при знаходженні елементу помітити його для вилучення

if(pr)

pr->\_del=-1;

return pr;

}

// корекція за двійковим пошуком

struct record \*updateBinary(struct keyString kArg, struct record \*pElm, struct record \*tb, int \*pQnElm)

{

// при знаходженні елементу заповнити його новими значеннями полів

struct record \*pr = selectBinary(kArg, tb, \*pQnElm);

pr->func = pElm->func;

pr->key = pElm->key;

pr->\_del = pElm->\_del;

return pr;

}

## vistab.h

void printRow(struct record \*);

void printTable(struct record \*, int, int);

## vistab.cpp

#include "stdafx.h"

#include "vistab.h"

#include "tables.h"

void printRow(struct record \*rw)

{

if(rw == 0)

printf("is absent\n");

else

if(rw->\_del)

printf("is deleted\n");

else

printf("%-10s %3u %5d\n", rw->key.str, rw->key.nMod, rw->func.\_f.i);

}

void printTable(struct record\* rw, int n0, int n1)

{

printf(" Table:\n");

while(n0 <= n1)

{

printRow(rw + n0);

n0++;

}

}

## stdafx.h

#pragma once

#include "targetver.h"

#include <stdio.h>

#include <tchar.h>

## stdafx.cpp

#include "stdafx.h"