НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №1

з дисципліни **«**Системне програмування 2**»**

Виконав:

студент 3 курсу

ФІОТ гр. ІО-21

Кузьменко Володимир

Перевірив:

Пустоваров В.І.

Київ – 2014 р.

## **Тема роботи**: **Модульне програмування в рамках базової системи проектування програм та його використання для побудови програм обробки таблиць**

Мета роботи: Вивчення типів таблиць в системних прог­ра­мах і конструкцій базової мови програмування для їх визначення. Пошук за прямою адресою. Основні типи залежностей та відно­шень, які реалізуються через пошук в таблицях системних програм. Лінійний та двійковий пошук. Вимоги до унікальності ключів.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № вар. | Тип ключа для прямої адреси | Тип ключа для інших видів пошуку | Тип функціо­нального поля | Тип вибірки |
| 12 | unsigned int | char\*\_ unsigned char | double | Черговий |

Лістинг програми

spLb1.cpp

#include "stdafx.h"

#include "tables.h"

#include "vistab.h"

#include <stdio.h>

int nTbStud=5, lnTbStud=100;

struct recrd tbStud[100]= // static table defintion

{{{"Diatlov",89},4,0}, // 1st record ititialization

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

{ {"Kuzmenko",56},7,0},

{ {"Chuchalo",12},9,0},

{ {"Konoz",32},4,0}

};// 2nd record ititializaton

struct recrd tstArg={{"Diatlov",89},41,0},

insArg={{"Gazizov",45},3,0},

insArg0={{"Kuzmenko",55},3,0},

insArg1={{"Kustar",4},3.5,0},

insArg2={{"Konoz",32},{4},0},

insArg3={{"Duchko",89},4.2,0},

insArg4={{"Suray",1},7,0},

insArg5={{"Chucha",8},9,0},

insArg6={{"Dovbenko",6},{4},0},

insArg7={{"Simonov",45},5.2,0},

insArg8={{"Urov",2},41,0};

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

{

prTab(tbStud, 0 , nTbStud-1);

printf("--------------------------------------------\n");

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

prRow(selNmb(tbStud, 1));

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

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

prTab(tbStud, 0, nTbStud-1);

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

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

prTab(tbStud, 0, nTbStud - 1);

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

delNmb(tbStud, 2);

prTab(tbStud, 0, nTbStud - 1);

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

printf("\nInsert linear: ");

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

prTab(tbStud, 0, nTbStud - 1);

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

prRow(selLin(tstArg.key, tbStud, 5));

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

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

prTab(tbStud, 0, nTbStud - 1);

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

prRow(updLin(insArg.key, &insArg4, tbStud, &nTbStud));

prTab(tbStud, 0, nTbStud - 1);

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

pckLin(tbStud, &nTbStud);

prTab(tbStud, 0, nTbStud - 1);

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

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

srtBin(tbStud, nTbStud);

prTab(tbStud, 0, nTbStud - 1);

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

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

prTab(tbStud, 0, nTbStud - 1);

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

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

prTab(tbStud, 0, nTbStud);

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

prRow(delBin(&insArg4, tbStud, &nTbStud));

prTab(tbStud, 0, nTbStud);

printf(" ");

return 0;

}

tables.h

struct keyStr// ключова частина запису

{char\* str; // ключові поля

unsigned char nMod;

};

// функціональна частина запису

struct fStr // functional part of record

{

double \_f;

};

struct recrd// структура рядка таблиці

{ // примірник структури ключа

struct keyStr key; // instantiation of key structure

// примірник функціональної частини

struct fStr func; // instantiation of functional part

// ознака вилучення

char \_del;

};

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

// 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 рядків

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

// порівняння за відношенням порядку для ключів

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

// порівняння за відношенням схожості

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

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

// вибірка за лінійним пошуком

// retreaving by linear search

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

// повторна вибірка за лінійним пошуком

// next retreaving by linear search

struct recrd\* selLinNxt(struct keyStr kArg,

struct recrd\* tb, int ln);

// включення за лінійним пошуком

// insertion by linear search

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

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

// вилучення за лінійним пошуком

// deleting by linear search

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

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

// корекція за лінійним пошуком

// correction by linear search

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

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

// пакування для лінійного пошуку

// packing for linear search

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

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

// сортування за двійковим пошуком

// sorting for linear search

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

// вибірка за двійковим пошуком

// retreaving by binary search

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

//---!!! реалізації дописуються за варіантом!!!---

// включення за двійковим пошуком

// insertion by binary search

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

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

// вилучення за двійковим пошуком

// deleting by binary search

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

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

// корекція за двійковим пошуком

// correction by binary search

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

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

tables.ccp  
#include "stdafx.h"

#include "tables.h"

#include <string.h>

// поржній елемент

struct recrd emptyElm= {{"",0}, // ключові поля

{0},0}; // функціональні поля

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

// 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)

{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 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)

{int n=\*pQnElm;

if(nElm>=n)return 0;

tb[nElm]=\*pElm;

return &tb[nElm];

}

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

// порівняння за відношенням нерівності

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

{return (strcmp(el->key.str, kArg.str)||

el->key.nMod != kArg.nMod);

}

// 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 для ключів

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;

}

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

// вибірка за лінійним пошуком

struct recrd\* selLin(struct keyStr kArg,

struct recrd\* tb, int ln)

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

if(ln<0)return 0;

return &tb[ln];

}

// повторна вибірка за лінійним пошуком

struct recrd\* selLinNxt(struct keyStr kArg,

struct recrd\* tb, int ln)

{static struct recrd\* prPtr=0;

struct recrd\* prPtrN;

if(prPtr==0)

prPtr=tb;

prPtrN=selLin(kArg, prPtr, ln);

if(prPtrN)prPtr=prPtrN+1;else prPtr=0;

return prPtrN;

}

// 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;

}

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

// 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

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;

}

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

struct recrd\*tb, int\*pQnElm)

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

// при знаходженні елементу помітити його для вилучення

if(pr)

pr->\_del=-1;

return pr;

}

// корекція за двійковим пошуком

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

struct recrd\*tb, int\*pQnElm)

{

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

// при знаходженні елементу заповнити його новими значеннями полів

pr->func = pElm->func;

pr->key = pElm->key;

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

return pr;

}

vistab.h

void prRow(struct recrd\* rw);

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

vistab.cpp

#include "stdafx.h"

#include "vistab.h"

#include "tables.h"

#include <stdio.h>

void prRow(struct recrd\* rw)

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

else if(rw->\_del!=0)printf("is deleted\n");

else printf("%-10s %3u %12u\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++;}

printf("\n");

}

stdafx.h

#pragma once

#include "targetver.h"

#include <stdio.h>

#include <tchar.h>

stdafx.cpp

#include "stdafx.h"