Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота № 1**

**з дисципліни**

**«Системне програмування»**

Виконала: студентка групи ІО-91

Косейкіна Ганна Сергіївна

2011 р.

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.

Залікова книжка №9123.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № 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 | |
| № вар. | Тип функції порівняння за мірою схожості | | | |
| 3 | Спiвпадiння максимальної кількості літер з врахуванням послідовності їх входження у ключі. | | | |

Лістінг:

SPLab1.cpp:

// SPLab1.cpp: определяет точку входа для консольного приложения.

#include "stdafx.h"

#include "tables.h"

#include "vistab.h"

#include <stdlib.h>

int nTb=7, lnTb=100;

struct recrd tb[100]= {{{"AaAaA", 7}, {4}, 0},

{{"Bbbsdb", 6}, {3}, 0},

{{"Ccsadkhw", 5}, {40}, 0},

{{"Esdf9", 4}, {32}, 0},

{{"Edfsd", 3}, {3}, 0},

{{"F3idsf", 2}, {41}, 0},

{{"Xzzzdf", 1}, {4}, 0}};

struct recrd InsNmb={{"Csdfhjh", 7}, {40}, 0},

SelLin={{"Csdfhjh", 7}, {40}, 0},

InsLin={{"Esdfdfs", 4}, {40}, 0},

DelLin={{"Bbbsdb", 6}, {3}, 0},

SelBin={{"Xzzzdf", 1}, {4}, 0},

InsBin={{"Zzzzzz", 4}, {3}, 0},

DelBin={{"Xzzzdf", 1}, {4}, 0},

UpdBin={{"Es3333", 4}, {3}, 0},

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

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

struct recrd\*pTb;

pTb = (struct recrd\*)malloc(100\*sizeof(struct recrd));

pTb = new struct recrd[100];

int t = 0;

printf("-----------selLab1----------\n");

prRow(selLab1(tb, SelLab1.key, nTb, &t));

printf("-----------selLab1----------\n");

prRow(selLab1(tb, SelLab1.key, nTb, &t));

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

prTab(tb, 0, nTb-1);

printf("-----------selNmb 0\n");

prRow(selNmb(tb, 0));

printf("-----------insNmb Csdfhjh 7 40\n");

insNmb(&InsNmb, tb, nTb, &nTb);

printf("-----------delNmb 2\n");

delNmb(tb, 2);

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

prTab(tb, 0, nTb-1);

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

pckNmb(tb, &nTb);

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

prTab(tb, 0, nTb-1);

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

srtNmb(tb, nTb);

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

prTab(tb, 0, nTb-1);

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

printf("-----------selLin Csdfhjh 7 40\n");

prRow(selLin(SelLin.key, tb, nTb, false));

printf("-----------insLin Esdfdfs 4 40\n");

insLin(&InsLin, tb, &nTb, false);

printf("-----------delLin AaAaA 7 4\n");

delLin(&DelLin, tb, &nTb, true);

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

prTab(tb, 0, nTb-1);

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

pckLin(tb, &nTb, false);

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

prTab(tb, 0, nTb-1);

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

srtLin(tb, nTb, true);

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

prTab(tb, 0, nTb-1);

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

prTab(tb, 0, nTb-1);

printf("-----------selBin Xzzzdf 1 4\n");

prRow(selBin(SelBin.key, tb, nTb, false));

printf("-----------insBin Zzzzzz 4 3\n");

insBin(&InsBin, tb, &nTb, false);

printf("-----------delBin Xzzzdf 1 4\n");

delBin(&DelBin, tb, &nTb,true);

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

prTab(tb, 0, nTb-1);

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

pckBin(tb, &nTb, false);

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

prTab(tb, 0, nTb-1);

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

srtBin(tb, nTb, true);

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

prTab(tb, 0, nTb-1);

printf("-----------updBin Zzzzzz 4 3 Es3333 4 3\n");

updBin(InsBin.key, &UpdBin, tb, &nTb, true);

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

prTab(tb, 0, nTb-1);

double a,b;

scanf("%lf%\*c%lf",&a,&b);

return 0;

}

stdafx.cpp:

// stdafx.cpp: исходный файл, содержащий только стандартные включаемые модули

// SPLab1.pch будет предкомпилированным заголовком

// stdafx.obj будет содержать предварительно откомпилированные сведения о типе

#include "stdafx.h"

stadfx.h:

// stdafx.h: включаемый файл для стандартных системных включаемых файлов

// или включаемых файлов для конкретного проекта, которые часто используются, но

// не часто изменяются

#pragma once

#include "targetver.h"

#include <stdio.h>

#include <tchar.h>

tables.cpp:

#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 char nElm){

#if ASM

int sr=4;

\_asm{

sub eax,eax

mov al,nElm

mov ecx,sr ;//sr=log2(SizeRc);

shl eax,cl

add eax,tb

}

#else

return &tb[nElm];

#endif

}

// включення за прямою адресою

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

}

void pckNmb(struct recrd\*tb, int\* ln){

int t = 0;

for(int i = 0; i < \*ln; i++)

if(tb[i].\_del == -1){

for(int j = i; j < \*ln; j++)

tb[j] = tb[j+1];

t++;

}

\*ln = \*ln - t;

}

void srtNmb(struct recrd\* tb, int ln){

for(int i = 0; i < ln-1; i++)

for(int j = i+1; j < ln; j++)

if(cmpKys(&(selNmb(tb, (unsigned char) i)->key), &selNmb(tb, (unsigned char) j)->key)== 1){

recrd t = tb[i];

tb[i] = tb[j];

tb[j] = t;

}

}

// вилучення за прямою адресою

struct recrd\* delNmb(struct recrd\*tb, unsigned char nElm){

tb[nElm].\_del=-1;

return &tb[nElm];

}

// корекція за прямою адресою

struct recrd\* updNmb(struct recrd\*pElm, struct recrd\*tb, unsigned char 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, bool keys){

if(keys)

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

else

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

}

// порівняння рядків

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

{unsigned n=0;

#if(ASM==0)

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

return s1[n]-s2[n];

#else

\_asm

{

xor eax,eax ; очистка акумулятора

; clear accumulator

xor ecx,ecx ; завантаження n

; loading n

mov esi,s1 ; dword ptr[ebp+8]; завантаження покажчика 1-го рядка

; loading pointer of the 1st string

mov edi,s2 ; dword ptr[ebp+12];завантаження покажчика 2-го рядка

; loading pointer of the 2nd string

lb: lodsb ; завантаження чергового байта 1-го рядка

; loading next byte of the 1st string

and al,al ; перевірка кінця 1-го рядка мови C/C++

; testing end of 1st string C/C++ language

jz lf ; на вихід з підпрограми

; to returning from this function

scasb ; порівняння чергових байтів рядків

; comparing next byte of strings

loopz lb ; продовження циклу порівняння

; continuing of comparison cycle

dec edi ; компенсація номера в рядку приймача

; compensation of 2nd string element number

lf: sub al,[edi]; формування значення відношення порядку

; forming value of order relation

cbw ; поширення знаку на 2 байти

; sign extension to 2 bytes

cwd ; поширення знаку на 4 байти

; sign extenіsion to 4 bytes

}

#endif

}

// 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;

}

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

if(keys){

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

if(i)

return i;

}

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

}

// порівняння за відношенням порядку для ключів

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

int i = 0;

while((k0->str[i] != 0) || (kArg->str[i] != 0)){

if(k0->str[i] > kArg->str[i])

return 1;

else if(k0->str[i] < kArg->str[i])

return -1;

i++;

}

if((k0->str[i] == 0)&&(kArg->str[i]==0))

if(k0->nMod < kArg->nMod)

return -1;

else if(k0->nMod > kArg->nMod)

return 1;

else

return 0;

else if (k0->str[i] == 0)

return -1;

return 1;

}

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

//------------------------------------------------------------------------------------struct recrd\* selLab1(struct recrd\* tb, struct keyStr kArg, int ln, int \*next){

int index = 0, n = 0;

if(\*next >= ln)

\*next = 0;

for(int i = \*next; i < ln; i++){

bool t = true;

int j = 0, n1 = 0;

while(t){

if(tb[i].key.str[j] != kArg.str[j]){

t = false;

n1--;

}

n1++;

j++;

}

if(n < n1){

index = i;

n = n1;

}

}

if(\*next > index){

int tnext = 0;

struct recrd\* temp = selLab1(tb, kArg, \*next-1, &tnext);

\*next = tnext;

return temp;

}

\*next = index+1;

return &tb[index];

}

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

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

void srtLin(struct recrd\* tb, int ln, bool keys){

for(int i = 0; i < ln-1; i++)

for(int j = i+1; j < ln; j++)

if(cmpKey(selLin(tb[i].key, tb, ln, keys), selLin(tb[j].key, tb, ln, keys)->key, keys) == 1){

recrd t = tb[i];

tb[i] = tb[j];

tb[j] = t;

}

}

// вибірка за лінійним пошуком

struct recrd\* selLin(struct keyStr kArg, struct recrd\* tb, int ln, bool keys){

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

if(ln<0)

return 0;

return &tb[ln];

}

// повторна вибірка за лінійним пошуком

struct recrd\* selLinNxt(struct keyStr kArg, struct recrd\* tb, int ln, bool keys){

static struct recrd\* prPtr=0;

struct recrd\* prPtrN;

if(prPtr==0)

prPtr=tb;

prPtrN=selLin(kArg, prPtr, ln, keys);

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, bool keys){

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, bool keys){

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

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, bool keys){

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

if(pEl)

\*pEl=\*pElm;

return pEl;

}

// пакування для лінійного пошуку

struct recrd\*pckLin(struct recrd\*tb, int\*pQnElm, bool keys){

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, bool keys){

int n=0, n1;

struct recrd el;

if(keys){

for(;n<ln;n++)

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

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

el=tb[n];

tb[n]=tb[n1];

tb[n1]=el;

}

}else{

for(;n<ln;n++)

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

if(tb[n].key.nMod > tb[n1].key.nMod){

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, bool keys){

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

recrd\* t = &emptyElm;

if(keys)

while(i=cmpKys(&tb[n].key, &kArg)){

if(i>0)

nU=n;

else

nD=n;

n=(nD+nU)>>1;

if(n==nD)

return t;

}

else

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

if(i>0)

nU=n;

else

nD=n;

n=(nD+nU)>>1;

if(n==nD)

return t;

}

return &tb[n];

}

// включення за двійковим пошуком

struct recrd\*insBin(struct recrd\*pElm, struct recrd\*tb, int\*pQnElm, bool keys){

short i, nD=-1, nU=\*pQnElm-1, n=(nD+nU)>>1;

while(i=cmpKey(&tb[n],pElm->key, keys)){

if(i>0)

nU=n;

else

nD=n;

n=(nD+nU)>>1;

// При відсутності елементу звільнити для нього місце,

if(n==nD){

(\*pQnElm)++;

// просунувши останні елементи в таблиці на одне заипс вперед

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

tb[i+1]=tb[i];

// Записати новий елемент на звільнене місце

tb[i+1]=\*pElm;

return &tb[i];

}

}

return &tb[n];

}

// вилучення за двійковим пошуком

void delBin(struct recrd\*pElm, struct recrd\*tb, int\*pQnElm, bool keys){

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

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

pr->\_del = -1;

}

// корекція за двійковим пошуком

struct recrd\*updBin(struct keyStr ks, struct recrd\*pElm, struct recrd\*tb, int\*pQnElm, bool keys){

struct recrd\*pr=selBin(ks, tb, \*pQnElm, keys);

pr->func= pElm->func;

pr->key= pElm->key;

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

// при знаходженні елементу заповнити його новими значеннями полів

return pr;

}

struct recrd\* pckBin(struct recrd\*tb, int\*pQnElm, bool keys){

int ln= \*pQnElm;

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

if(selBin(tb[i].key, tb, ln, keys)->\_del==-1){

for(int j = i; j < ln; j++)

tb[j] = tb[j+1];

ln--;

}

}

\*pQnElm=ln;

return tb;

}

tables.h:

struct f{

int i;

};

struct keyStr// ключова частина запису

{char\* str; // ключові поля

unsigned long nMod;

};

// функціональна частина запису

struct fStr

{

struct f \_f; };

struct recrd// структура рядка таблиці

{ // примірник структури ключа

struct keyStr key;

// примірник функціональної частини

struct fStr func;

// ознака вилучення

char \_del; };

#define ASM 1 // без вставок на Асемблері, коли 0

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

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

// обробка записів таблиць за прямою адресою

// вибірка за прямою адресою

void srtNmb(struct recrd\* tb, int ln);

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

// включення за прямою адресою

void pckNmb(struct recrd\*tb, int \*ln);

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

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

// вилучення за прямою адресою

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

// корекція за прямою адресою

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

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

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

//------------------------------------------------------------------------------------// порівняння за відношенням нерівності

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

// 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);

struct recrd\* selLab1(struct recrd\* tb, struct keyStr key, int ln, int \*next);

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

// вибірка за лінійним пошуком

void srtLin(struct recrd\* tb, int ln, bool keys);

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

// повторна вибірка за лінійним пошуком

struct recrd\* selLinNxt(struct keyStr kArg, struct recrd\* tb, int ln, bool keys);

// включення за лінійним пошуком

struct recrd\*insLin(struct recrd\*pElm, struct recrd\*tb, int\*pQnElm, bool keys);

// вилучення за лінійним пошуком

struct recrd\*delLin(struct recrd\*pElm, struct recrd\*tb, int\*pQnElm, bool keys);

// корекція за лінійним пошуком

struct recrd\*updLin(struct keyStr, struct recrd\*pElm, struct recrd\*tb, int\*pQnElm, bool keys);

// пакування для лінійного пошуку

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

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

// сортування за двійковим пошуком

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

// вибірка за двійковим пошуком

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

// включення за двійковим пошуком

struct recrd\*insBin(struct recrd\*pElm, struct recrd\*tb, int\*pQnElm, bool keys);

// вилучення за двійковим пошуком

void delBin(struct recrd\*pElm, struct recrd\*tb, int\*pQnElm, bool keys);

// корекція за двійковим пошуком

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

struct recrd\*pckBin(struct recrd\*tb, int\*pQnElm, bool keys);

vistab.cpp:

#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!=0)printf("is deleted\n");

else printf("%-10s %3u %10u\n",

rw->key.str,rw->key.nMod,rw->func.\_f.i);

}

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

{printf(" Table:\n");

while(n0<=n1){prRow(rw+n0);n0++;}

printf("\n");

}

vistab.h:

enum color

{violet, blue, green, yellow, orange, red

};

void prClr(enum color cl);

void prRow(struct recrd\* rw);

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