**Лабораторная работа №14. Хэш-таблицы с цепочками**

**Вариант 10**

Условие:

**Школа**. Создать хэш-таблицу со следующими полями: номер школы, фамилия директора. Ключ – номер школы.

Код:

Главная функция:

ConsoleApplication 27,05.cpp

#include "stdafx.h"

#include "Hash\_Twin\_Chain.h"

#include <iostream>

using namespace std;

struct AAA

{ int key;

char \*mas;

AAA(int k, char\*z)

{

key = k;

mas = z;

}

AAA()

{

key = 0;

mas = "";

}

};

int hf(void\* d)//получение ключа

{ AAA\* f = (AAA\*)d;

return f->key;

}

void AAA\_print(listx::Element\* e)

{

std::cout<<((AAA\*)e->Data)->key<<'-'<<((AAA\*)e->Data)->mas<<" / ";

}

int \_tmain(int argc, \_TCHAR\* argv[])

{ setlocale(LC\_ALL, "rus");

int current\_size =7;

cout<<"Ведите размер хэш-таблицы"<<endl;

cin>>current\_size;

hashTC::Object H = hashTC::Create(current\_size, hf);

int choise; int k;

for(;;)

{

cout<<"1 - вывод хэш-таблицы"<<endl;

cout<<"2 - добавление элемента"<<endl;

cout<<"3 - удаление элемента"<<endl;

cout<<"4 - поиск элемента"<<endl;

cout<<"0 - выход"<<endl;

cout<<"сделайте выбор"<<endl;

cin>>choise;

switch(choise)

{

case 0: exit(0);

case 2: {

AAA \*a = new AAA;

char \*str = new char [20];

cout<<"введите номер школы"<<endl; cin>>k;

a->key = k;

cout<<"введите город"<<endl;

cin>>str;

a->mas = str;

H.Insert(a);

break;

}

case 1:

H.Scan();

break;

case 3: {

AAA \*b = new AAA;

cout<<"введите ключ"<<endl;

cin>>k;

b->key = k;

H.Delete(b);

break;}

case 4: {

AAA \*c = new AAA;

cout<<"введите ключ"<<endl;

cin>>k;

c->key = k;

if (H.Search(c) == NULL)

cout<<"Элемент не найден"<<endl;

else {

cout<<"Первый элемент с данным клю-чом"<<endl;

AAA\_print(H.Search(c)); cout<<endl; }

break;

}

}

}

return 0;

}

Hash\_ Table\_(Twin\_Chain).cpp

#include "stdafx.h"

#include "Hash\_Twin\_Chain.h"

#include "Lists.h"

#include <iostream>

struct AAA

{ int key;

char \*mas;

AAA(int k, char\*z)

{ key = k; mas = z; }

};

namespace hashTC

{

Object Create(int size,int (\*f)(void\*))

{

return \*(new Object(size, f));

}

int Object::HashFunction(void\* data)

{

return (FunKey(data) % Size);

};

bool Object::Insert(void\* data)

{ return (Hash[HashFunction(data)].Insert(data)); };

bool Object::Delete(void\* data)

{ return (Hash[HashFunction(data)].Delete(data)); };

listx::Element\* Object::Search(void\* data)

{ return Hash[HashFunction(data)].Search(data); };

void Object::Scan()

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

{ Hash[i].Scan(); std::cout<<'\n';}

};

}

Lists.cpp

#include "stdafx.h"

#include "Lists.h"

#include <iostream>

struct AAA //элемент таблицы

{ int key;

char \*mas;

};

namespace listx

{ bool Object::Insert(void\* data)

{ bool rc = NULL;

if(Head == NULL) Head = new Element(NULL, data, Head);

else Head = (Head->Prev = new Element(NULL, data, Head));

return rc;

}

Element\* Object::Search(void\* data)

{ Element\* rc=Head;

while((rc != NULL) && ((((AAA\*)rc->Data)->key) != ((AAA\*)data)->key))

rc = rc->Next;

return rc;

}

bool Object::Delete(Element\* e)

{ bool rc = NULL;

if(rc = (e != NULL))

{ if(e->Next != NULL) e->Next->Prev = e->Prev;

if(e->Prev != NULL) e->Prev->Next = e->Next;

else Head = e->Next;

delete e;

}

std::cout<<"Элемент удален"<<std::endl;

return rc;

}

bool Object::Delete(void\* data)

{ return Delete(Search(data)); }

Element\* Object::GetLast()

{ listx::Element\* e = this->GetFirst(), \*rc = this->GetFirst();

while(e != NULL)

{ rc = e; e = e->GetNext(); };

return rc;

}

Object Create()

{ return \*(new Object()); };

void Object::Scan() // просмотр

{ listx::Element\* e = this->GetFirst();

while(e != NULL)

{ std::cout<<((AAA\*)e->Data)->key<<'-'<<((AAA\*)e->Data)->mas<<" / ";

e = e->GetNext();

};

}

}

Lists.h

#pragma once //Заголовочный файл List.h

#define LISTNIL (Element\*)-1

namespace listx

{ struct Element

{ Element\* Prev;

Element\* Next;

void\* Data;

Element(Element\* prev, void\* data, Element\* next)

{ Prev = prev; Data = data; Next = next; }

Element\* GetNext()

{ return Next; };

Element\* GetPrev()

{ return Prev; };

};

static Element\* NIL = NULL;

struct Object

{

Element\* Head;

Object() {

Head = NIL;

};

Element\* GetFirst() { return Head; };

Element\* GetLast(); Element\* Search(void\* data);

bool Insert(void\* data); bool Delete(Element\* e);

bool Delete(void\* data); void Scan();

};

Object Create();

}

#undef LISTNIL

Hash\_Twin\_Chain.h

#pragma once //Заголовочный файл Hash\_ Twin\_Chain.h

#include "Lists.h"

namespace hashTC

{ struct Object

{

int Size;

int (\*FunKey)(void\*);

listx::Object\* Hash;

Object(int size, int (\*f)(void\*))

{ Size = size;

FunKey = f;

Hash = new listx::Object[Size];

};

int HashFunction(void\* data);

bool Insert(void\* data);

listx::Element\* Search(void\* data);

bool Delete(void\* data);

void Scan();

};

Object Create(int size, int (\*f)(void\*));

}