**5 ЛАБОРАТОРНАЯ РОБОТА №5**

**КЕШУВАННЯ ПАМ’ЯТІ**

**5.1 Мета роботи**

Ознайомитись з алгоритмами кешування, визначити переваги алгоритмів та порівняти їх, розробити власний програмний застосунок.

**5.2 Виконання роботи**

#ifndef CACHE

#define CACHE

#include <QMap>

#include <QSharedPointer>

#include <time.h>

#include <QDebug>

template<typename TKey, typename TValue>

class Cache {

private:

typedef QSharedPointer<TValue> DataPtr;

class Entry {

DataPtr **pData**;

double **frequency**;

long **cnt**;

clock\_t **timeStart**;

bool **isTake** = false;

public:

Entry(TValue \*ptrData) {

**pData**.reset(ptrData);

**timeStart** = clock();

**cnt** = 0;

}

TValue \*get() {

**cnt** ++;

**isTake** = true;

return **pData**.data();

}

double getFrequency() {

if(**isTake**) {

clock\_t now = clock();

**frequency** = (double)**cnt** / (double)(now - **timeStart**);

}

return **frequency**;

}

};

QMap<TKey, Entry\*> **mapValues**;

int **size** = 0;

public:

Cache(int size = 10) {

this->**size** = size;

}

void add(TKey key, TValue \*ptrData) {

if(**size** == **mapValues**.size())

chakedAndRemoveOld();

**mapValues**.insert(key, new Entry(ptrData));

}

TValue \*get(TKey key) {

Entry \*ptr = **mapValues**.value(key);

if(ptr != NULL) {

return ptr->get();

}

return NULL;

}

void show() {

qDebug() << "Show: size = " << **mapValues**.size() ;

typename QMap<TKey, Entry\*>::iterator st = **mapValues**.begin();

for(; st != **mapValues**.end(); st ++) {

qDebug() << "key:" << st.key() << st.value()->getFrequency() ;

}

}

private:

void chakedAndRemoveOld() {

typename QMap<TKey, Entry\*>::iterator st = **mapValues**.begin();

typename QMap<TKey, Entry\*>::iterator old = st;

st ++;

for(; st != **mapValues**.end(); st ++) {

if(old.value()->getFrequency() > st.value()->getFrequency()) {

old = st;

}

}

**mapValues**.remove(old.key());

}

};

#endif // CACHE

#include <QCoreApplication>

#include "cache.h"

class DataT

{

private:

int **data** = 0;

public:

DataT(int data) {

this->**data** = data;

}

};

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

{

QCoreApplication a(argc, argv);

Cache<int, DataT> cashe(10);

int \*ptrCntGet = new int[10];

for(int i = 0; i < 10; i ++) {

cashe.add(i, new DataT(i));

ptrCntGet[i] = 20 - i;

}

for( int i = 0; i < 10; i++ ) {

for( int k = 0; k < ptrCntGet[i]; k ++ ) {

cashe.get(i);

}

}

cashe.show();

cashe.add(11, new DataT(11));

cashe.show();

return a.exec();

}

**5.3 Висновки**

У ході виконання даної лабораторної роботи ознайомились з алгоритмами кешування, визначили переваги алгоритмів та порівняли їх, розробили власний програмний застосунок.