Лабораторная работа 12

По дисциплине «Основы алгоритмизации и программирования»

На тему «Бинарные кучи»

***Задание#1***

*Macros.h*

#include <iostream>

#include <vector>

#include <ctime>

#include <cmath>

#include <iomanip>

#include <stack>

#include <map>

#include <cstring>

#include <string>

#include <chrono>

#include <fstream>

#include <Windows.h>

#include <queue>

#define vi cout <<

#define vv cin >>

#define nl cout << "\n";

#define nw cout << "\t";

#define SCOCP1251 SetConsoleOutputCP(1251);

#define SCCP1251 SetConsoleCP(1251);

#define ret return

#define pause system("pause");

using namespace :: std;

*.cpp*

#include "Macros.h"

// Структура для узла бинарного дерева

struct AAA {

int x;

void print();

};

void AAA::print() {

vi x;

}

namespace heap {

enum CMP { LESS = -1, EQUAL = 0, GREAT = 1 };

struct Heap {

int size;

int maxSize;

void\*\* storage;

CMP(\*compare)(void\*, void\*);

Heap(int maxsize, CMP(\*f)(void\*, void\*)) {

size = 0;

storage = new void\* [maxSize = maxsize];

compare = f;

};

int left(int ix);

int right(int ix);

int parent(int ix);

bool isFull() const {

ret(size >= maxSize);

};

bool isEmpty() const {

ret(size <= 0);

};

bool isLess(void\* x1, void\* x2) const {

ret compare(x1, x2) == LESS;

};

bool isGreat(void\* x1, void\* x2) const {

ret compare(x1, x2) == GREAT;

};

bool isEqual(void\* x1, void\* x2) const {

ret compare(x1, x2) == EQUAL;

};

void swap(int i, int j);

void heapify(int ix);

void insert(void\* x);

void\* extractMax();

void\* extractMin(); // добавлено

void\* extractI(int i); // добавлено

void unionHeap(Heap& h); // добавлено

void scan(int i) const;

};

Heap create(int maxsize, CMP(\*f)(void\*, void\*));

Heap create(int maxsize, CMP(\*f)(void\*, void\*)) {

ret\* (new Heap(maxsize, f));

}

int Heap::left(int ix) {

ret(2 \* ix + 1 >= size) ? -1 : (2 \* ix + 1);

}

int Heap::right(int ix) {

ret(2 \* ix + 2 >= size) ? -1 : (2 \* ix + 2);

}

int Heap::parent(int ix) {

ret(ix + 1) / 2 - 1;

}

void Heap::swap(int i, int j) {

void\* buf = storage[i];

storage[i] = storage[j];

storage[j] = buf;

}

void Heap::heapify(int ix) {

int l = left(ix), r = right(ix), irl = ix;

if (l > 0) {

if (isGreat(storage[l], storage[ix])) irl = l;

if (r > 0 && isGreat(storage[r], storage[irl])) irl = r;

if (irl != ix) {

swap(ix, irl);

heapify(irl);

}

}

}

void Heap::insert(void\* x) {

int i;

if (!isFull()) {

storage[i = ++size - 1] = x;

while (i > 0 && isLess(storage[parent(i)], storage[i])) {

swap(parent(i), i);

i = parent(i);

}

}

}

void\* Heap::extractMax() {

void\* rc = nullptr;

if (!isEmpty()) {

rc = storage[0];

storage[0] = storage[size - 1];

size--;

heapify(0);

}

ret rc;

}

void\* Heap::extractMin() {

void\* rc = nullptr;

if (!isEmpty()) {

rc = storage[size - 1];

storage[size - 1] = storage[0];

size--;

heapify(0);

}

ret rc;

}

void\* Heap::extractI(int i) {

void\* rc = nullptr;

if (!isEmpty() && i >= 0 && i < size) {

rc = storage[i];

storage[i] = storage[size - 1];

size--;

heapify(0);

}

ret rc;

}

void Heap::unionHeap(Heap& h) {

for (int i = 0; i < h.size; ++i) {

insert(h.storage[i]);

}

}

void Heap::scan(int i) const {

int probel = 20;

nl

if (size == 0)

vi "Куча пуста";

for (int u = 0, y = 0; u < size; u++) {

vi setw(probel + 10) << setfill(' ');

((AAA\*)storage[u])->print();

if (u == y) {

nl

if (y == 0)

y = 2;

else

y += y \* 2;

}

probel /= 2;

}

nl

}

}

heap::CMP cmpAAA(void\* a1, void\* a2) {

#define A1 ((AAA\*)a1)

#define A2 ((AAA\*)a2)

heap::CMP rc = heap::EQUAL;

if (A1->x > A2->x)

rc = heap::GREAT;

else

if (A2->x > A1->x)

rc = heap::LESS;

ret rc;

#undef A2

#undef A1

}

int main() {

SCOCP1251

SCCP1251

int k, choice;

heap::Heap h1 = heap::create(30, cmpAAA);

heap::Heap h2 = heap::create(30, cmpAAA); // дополнительная куча для объединения

for (;;) {

vi "1 - вывод кучи на экран"; nl

vi "2 - добавить элемент"; nl

vi "3 - удалить максимальный элемент"; nl

vi "4 - удалить минимальный элемент"; nl // добавлено

vi "5 - удалить i-ый элемент"; nl // добавлендо

vi "6 - объединить две кучи"; nl // добавлено

vi "0 - выход"; nl

vi "choice?"; nl

vv choice;

switch (choice) {

case 0:

exit(0);

case 1:

h1.scan(0);

break;

case 2:

{

AAA\* a = new AAA;

vi "ключ?"; nl

vv k;

a->x = k;

h1.insert(a);

}

break;

case 3:

h1.extractMax();

break;

case 4:

h1.extractMin();

break;

case 5:

vi "индекс элемента для удаления?"; nl

vv k;

h1.extractI(k);

break;

case 6:

h1.unionHeap(h2);

break;

default:

nl vi "неверная команда!"; nl

}

}

pause

ret 0;

}



