Беларускі дзяржаўны тэхналагічны ўніверсітэт

Факультэт інфармацыйных тэхналогій

Кафедра праграмнай інжэнерыі

Лабараторная работа №12

Па дысцыпліне «Асновы алгарытмізацыі і праграмавання»

На тэму «Бінарныя кучы»

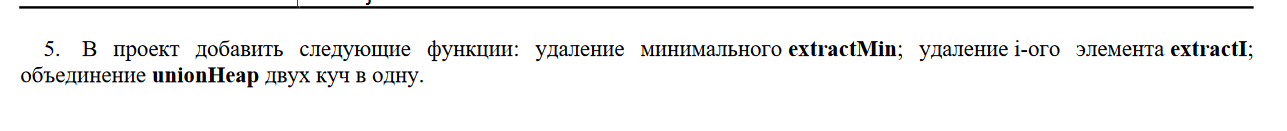
Выканала:

Студэнтка 1 курса 6 группы

Жучкевіч Кацярына Сяргееўна

Выкладчык: асс. Андронава М.В.

Мінск, 2024



Разам з дадатковымі варыянтамі

Heap.cpp

#include "heap.h"

#include <iostream>

#include <iomanip>

void AAA::print()

{

std::cout << x;

}

namespace heap

{

Heap create(int maxsize, CMP(\*f)(void\*, void\*))//создание кучи

{

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

}

int Heap::left(int ix)//левая часть

{

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

}

int Heap::right(int ix)//правая часть

{

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

}

int Heap::parent(int ix)

{

return (ix + 1) / 2 - 1;

}

void Heap::swap(int i, int j)//обмен элементов

{

void\* buf = storage[i];

storage[i] = storage[j];

storage[j] = buf;

}

void Heap::heapify(int ix)//восстановление кучи

{

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)//вставка элемента

{

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()//извлечь максимальный

{

void\* rc = nullptr;

if (!isEmpty())

{

rc = storage[0];

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

size--;

heapify(0);

} return rc;

}

void Heap::scan(int i) const //Вывод значений элементов на экран

{

int probel = 20;

std::cout << '\n';

if (size == 0)

std::cout << "Куча пуста";

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

{

std::cout << std::setw(probel + 10) << std::setfill(' ');

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

if (u == y)

{

std::cout << '\n';

if (y == 0)

y = 2;

else

y += y \* 2;

}

probel /= 2;

}

std::cout << '\n';

}

void\* Heap::extractMin() {

void\* rc = nullptr;

if (!isEmpty()) // праглядаем, каб у кучы штосьці было

{

int minIndex = 0; //мін значэнне = 0

for (int i = 1; i < size; ++i) {//цыкл для пошуку

if (isLess(storage[i], storage[minIndex])) { //шукаем мін элемент

minIndex = i; //запісваем яго індэкс

}

}

rc = storage[minIndex]; //захоўваем яго індэкс

storage[minIndex] = storage[size - 1]; //змяняем месцамі

size--; //і памяншаем памер

heapify(minIndex);

}

return rc; //вяртаем значэнне індэкса

}

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

void\* rc = nullptr;

if (!isEmpty()) { //праглядаем, каб у кучы штосьці было

swap(i, size - 1);//змяняем элементы месцамі

size--; //памяншаем памер кучы

heapify(i);

}

return rc;

}

void Heap::unionHeap(Heap& h2) {

for (int i = 0; i < h2.size; ++i) //праглядаем другую кучу

{

insert(h2.storage[i]); //устаўляем элементы

}

h2.size = 0; //чысцім

}

int Heap::countNegative()//лічым колькасць элементаў, якія больш за 3

{

int counter = 0;

for (int i = 0; i < size; i++)

{

if (!(((AAA\*)storage[i])->compare()))

{

counter++;

}

}

return counter;

}

int Heap::countPositive()//лічым колькасць адмоўных элементаў

{

int counter = 0;

for (int i = 0; i < size; i++)

{

if (!(((AAA\*)storage[i])->compare()))

{

counter++;

}

}

return counter;

}

void Heap::findMnM(Heap& second)//змяняем месцамі мах і мін элементы куч

{

int maxIndex1 = 0;

int minIndex2 = 0;

for (int i = 1; i < size; i++)//знаходзім гэтыя элементы

{

if (isGreat(storage[i], storage[maxIndex1]))

{

maxIndex1 = i;

}

}

for (int i = 1; i < second.size; i++)

{

if (second.isLess(second.storage[i], storage[minIndex2]))

{

minIndex2 = i;

}

}

swap(maxIndex1, minIndex2);

heapify(maxIndex1);

second.heapify(minIndex2);

}

}

Main.cpp

#include "Heap.h"

#include <iostream>

#include <Windows.h>

using namespace std;

heap::CMP cmpAAA(void\* a1, void\* a2) //Функция сравнения

{

#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;

return rc;

#undef A2

#undef A1

}

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

int main()

{

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

setlocale(LC\_ALL, "rus");

int k, choice;

int elem;

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

for (;;)

{

cout << "1 - вывад кучы" << endl;

cout << "2 - дадаць элемент" << endl;

cout << "3 - выдаліць max" << endl;

cout << "4 - выдаліць min" << endl;

cout << "5 - палічыць колькасць адмоўных элементаў" << endl;

cout << "6 - палічыць колькасць станоўчых элементаў\n";

cout << "7 - выдаліць элемент\n";

cout << "0 - выход\n" << endl;

cout << "Зрабіце выбар" << endl; cin >> choice;

switch (choice)

{

case 0: exit(0);

case 1: h1.scan(0);

break;

case 2: { AAA\* a = new AAA;

cout << "введите ключ" << endl; cin >> k;

a->x = k;

h1.insert(a);

}

break;

case 3: h1.extractMax(); break;

case 4: h1.extractMin(); break;

case 5: cout << "\nКолькасць элементаў меньш за 0\n" << h1.countNegative() << endl; break;

case 6: cout << "\nКолькасць элементаў меньш за 0\n" << h1.countNegative() << endl; break;

case 7: cout << "\nУвядзіце: "; cin >> elem; h1.extractI(elem);

default: cout << endl << "Упс!" << endl;

}

} return 0;

}

Heap.h

#pragma once

struct AAA

{

int x;

void print();

bool compare()

{

return x > 0;

}

};

namespace heap

{

enum CMP//для сравнения

{

LESS = -1, EQUAL = 0, GREAT = 1

};

struct Heap

{

int size;

int maxSize;

void\*\* storage; // данные

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

{

return (size >= maxSize);

};

bool isEmpty() const

{

return (size <= 0);

};

bool isLess(void\* x1, void\* x2) const//функции сравнения

{

return compare(x1, x2) == LESS;

};

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

{

return compare(x1, x2) == GREAT;

};

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

{

return compare(x1, x2) == EQUAL;

};

void swap(int i, int j);

void heapify(int ix);

void insert(void\* x);

void\* extractMax();

void scan(int i) const;

void\* extractMin();

void\* extractI(int i);

void unionHeap(Heap& second);

int countNegative();

int countPositive();

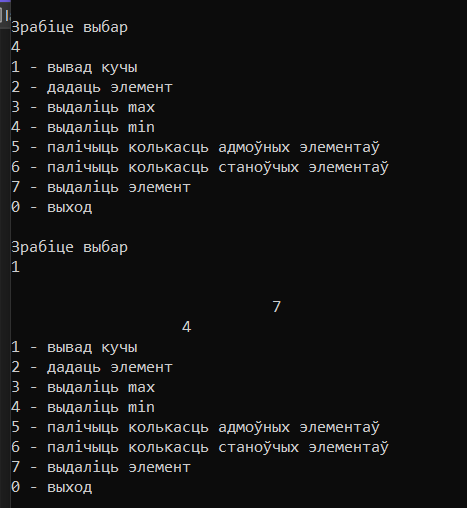
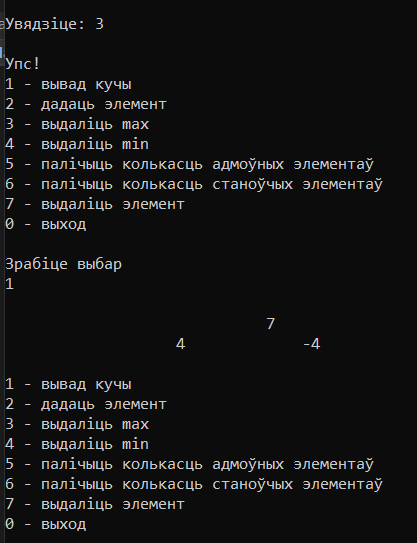
void findMnM(Heap& second);

};

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

};

Асноўныя:



Дадатковыя:

