# Учреждение образования

# «Белорусский государственный технологический университет»

Кафедра информационных систем и технологий

Отчёт по лабораторной работе №13

Вариант 2

Выполнил:

студент 1 курса 2 группы Самцевич Алексей

Минск 2020

Задание №1: .  В проект добавить следующие функции: удаление минимального **extractMin**; удаление i-ого элемента **extractI**; объединение **unionHeap** двух куч в одну.

Код:

Главный файл:

//

// main.cpp

// laba\_13.1

//

// Created by Alex Samtsevich on 08.06.2020.

// Copyright © 2020 Alex Samtsevich. All rights reserved.

//

#include <iostream>

#include "Heap.hpp"

using namespace std;

heap::CMP cmpAAA(void\* a1, void\* a2) //‘ункци€ сравнени€

{

#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()

{

setlocale(LC\_ALL, "rus");

int k, choice;

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

heap::Heap h2 = heap::create(30, cmpAAA);

AAA\* b1 = new AAA; b1->x = 12; h2.insert(b1);

AAA\* b2 = new AAA; b2->x = 3; h2.insert(b2);

AAA\* b3 = new AAA; b3->x = 5; h2.insert(b3);

AAA\* b4 = new AAA; b4->x = 16; h2.insert(b4);

AAA\* b5 = new AAA; b5->x = 9; h2.insert(b5);

for (;;)

{

cout << "1 - вывод кучи на экран" << endl;

cout << "2 - добавить элемент" << endl;

cout << "3 - удалить максимальный элемент" << endl;

cout << "4 - удалить минимальный элемент" << endl;

cout << "5 - удалить элемент по индексу" << endl;

cout << "6 - слияние двух куч" << endl;

cout << "0 - выход" << endl;

cout << "сделайте выбор" << 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:

int index;

cout << "введите индекс:" << endl;

cin >> index;

h1.extractForIndex(index);

break;

case 6:

h1.marger(h2);

break;

default: cout << endl << "введена неверная команда!" << endl;

}

} return 0;

}

Head.cpp:

//

// Heap.cpp

// laba\_13.1

//

// Created by Alex Samtsevich on 08.06.2020.

// Copyright © 2020 Alex Samtsevich. All rights reserved.

//

#include "Heap.hpp"

#include <iostream>

#include <iomanip>

void AAA::print()

{

std::cout << x;

}

int AAA::getPriority() const

{

return x;

}

namespace heap

{

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

{

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

}

int Heap::left(int ix)

{

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

}

int Heap::right(int ix)

{

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)

{

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);

} 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())

{

int min = 0;

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

{

if (((AAA\*)storage[i])->x < ((AAA\*)storage[min])->x)min = i;

}

rc = storage[min];

if (size - 1 != min)storage[min] = storage[size - 1];

else storage[min] = storage[size - 2];

size--;

heapify(0);

} return rc;

}

void\* Heap::extractForIndex(int ix)

{

void\* rc = nullptr;

if (!isEmpty())

{

rc = storage[ix];

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

size--;

heapify(0);

} return rc;

}

void Heap::marger(Heap b)

{

/\*function merge(a, b : Heap):

for i = 0 to b.heapSize - 1

a.heapSize = a.heapSize + 1

a[a.heapSize - 1] = b[i]

a.heapify()\*/

int s = size;

for (int i = 0; i < b.size; ++i, ++s)

{

size++;

storage[s] = b.storage[i];

}

heapify(0);

Head.hpp:

//

// Heap.hpp

// laba\_13.1

//

// Created by Alex Samtsevich on 08.06.2020.

// Copyright © 2020 Alex Samtsevich. All rights reserved.

//

#pragma once

#include <iostream>

struct AAA

{

int x;

void print();

int getPriority() const;

};

namespace heap

{

enum CMP

{

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

{

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\* extractMin();

void\* extractForIndex(int ix);

void marger(Heap b);

void scan(int i) const;

};

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

};

}

Скриншот:



