Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа 11

По дисциплине «Основы алгоритмизации программирования»

На тему «Бинарные деревья»

Выполнил:

Студент 1 курса 6 группы

Кучерук Николай Петрович

Преподаватель: асс. Андронова М.В.

2024, Минск

**Задание**

5.  В проект добавить следующие функции: удаление минимального **extractMin**; удаление i-ого элемента **extractI**; объединение **unionHeap** двух куч в одну.

**Дополнительные задания**

1. Функция, суммирующая все элементы кучи

2. Удаление элемента по значению из кучи

3. Функция, для очистки кучи

|  |
| --- |
| **Код программы** |
| **Heap.h**  #pragma once  #include <iostream>  #include <iomanip>  // Определение структуры AAA, которая будет использоваться в качестве элемента кучи  struct AAA  {  int x; // Значение элемента  void print() // Функция для вывода значения элемента  {  std::cout << x;  }  };  namespace heap  {  // Перечисление для сравнения элементов кучи  enum CMP  {  LESS = -1, // Элемент меньше  EQUAL = 0, // Элементы равны  GREAT = 1 // Элемент больше  };  // Основной класс кучи  struct Heap  {  int size; // Текущий размер кучи  int maxSize; // Максимальный размер кучи  void\*\* storage; // Указатель на массив элементов кучи  CMP(\*compare)(void\*, void\*); // Функция сравнения элементов  // Конструктор кучи  Heap(int maxsize, CMP(\*f)(void\*, void\*));  // Деструктор кучи  ~Heap();  // Функции для работы с индексами элементов кучи  int left(int ix); // Индекс левого потомка  int right(int ix); // Индекс правого потомка  int parent(int ix); // Индекс родителя  // Проверки состояния кучи  bool isFull() const; // Куча полна  bool isEmpty() const; // Куча пуста  bool isLess(void\* x1, void\* x2) const; // Первый элемент меньше второго  bool isGreat(void\* x1, void\* x2) const; // Первый элемент больше второго  bool isEqual(void\* x1, void\* x2) const; // Элементы равны  // Операции с элементами кучи  void swap(int i, int j); // Обмен элементов  void heapify(int ix); // Преобразование кучи в кучу  void insert(void\* x); // Вставка элемента  void\* extractMax(); // Извлечение максимального элемента  void\* extractMin(); // Извлечение минимального элемента  void extractI(int i); // Извлечение элемента по индексу  void merge(Heap& h2); // Объединение двух куч  void sum(); // Суммаа элементов  void remove(void\* value); // Удаление элемента  void clear(); // Очистка кучи  void scan(int i) const; // Вывод содержимого кучи  };  // Создание новой кучи  Heap create(int maxsize, CMP(\*f)(void\*, void\*));  }  **Heap.cpp**  #include "Heap.h"  #include <iostream>  namespace heap  {  // Конструктор кучи, инициализирует размер, максимальный размер и функцию сравнения  Heap::Heap(int maxsize, CMP(\*f)(void\*, void\*))  {  size = 0;  maxSize = maxsize;  storage = new void\* [maxSize];  compare = f;  }  // Деструктор кучи, освобождает память, выделенную под хранилище  Heap::~Heap()  {  delete[] storage;  }  // Получение индекса левого потомка для заданного индекса  int Heap::left(int ix)  {  return (2 \* ix + 1 >= size) ? -1 : (2 \* ix + 1);  }  // Получение индекса правого потомка для заданного индекса  int Heap::right(int ix)  {  return (2 \* ix + 2 >= size) ? -1 : (2 \* ix + 2);  }  // Получение индекса родителя для заданного индекса  int Heap::parent(int ix)  {  return (ix + 1) / 2 - 1;  }  // Проверка, полна ли куча  bool Heap::isFull() const  {  return (size >= maxSize);  }  // Проверка, пуста ли куча  bool Heap::isEmpty() const  {  return (size <= 0);  }  // Сравнение двух элементов: первый меньше второго  bool Heap::isLess(void\* x1, void\* x2) const  {  return compare(x1, x2) == LESS;  }  // Сравнение двух элементов: первый больше второго  bool Heap::isGreat(void\* x1, void\* x2) const  {  return compare(x1, x2) == GREAT;  }  // Сравнение двух элементов: элементы равны  bool Heap::isEqual(void\* x1, void\* x2) const  {  return compare(x1, x2) == EQUAL;  }  // Обмен двух элементов в куче  void Heap::swap(int i, int j)  {  void\* buf = storage[i];  storage[i] = storage[j];  storage[j] = buf;  }  // Преобразование кучи в кучу, начиная с заданного индекса  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);  }  }  }  // Вставка элемента в кучу  void Heap::insert(void\* x)  {  if (!isFull())  {  storage[size++] = x;  int i = size - 1;  while (i > 0 && isLess(storage[parent(i)], storage[i]))  {  swap(parent(i), i);  i = parent(i);  }  }  }  // Извлечение максимального элемента из кучи  void\* Heap::extractMax()  {  void\* rc = nullptr;  if (!isEmpty())  {  rc = storage[0];  storage[0] = storage[size - 1];  size--;  heapify(0);  }  return rc;  }  // Извлечение элемента по индексу из кучи  void Heap::extractI(int i)  {  if (i < 0 || i >= size)  {  std::cerr << "Ошибка." << std::endl;  return;  }  swap(i, size - 1);  size--;  heapify(i);  }  // Объединение двух куч  void Heap::merge(Heap& h2)  {  for (int i = 0; i < h2.size; ++i)  {  insert(h2.storage[i]);  }  h2.size = 0;  }  // Удаление элемента из кучи  void Heap::remove(void\* value)  {  for (int i = 0; i < size; ++i)  {  if (isEqual(storage[i], value))  {  extractI(i);  return;  }  }  std::cerr << "Ошибка." << std::endl;  }  // Очистка кучи  void Heap::clear()  {  size = 0;  }  // Создание новой кучи  Heap create(int maxsize, CMP(\*f)(void\*, void\*))  {  return Heap(maxsize, f);  }  // Вывод содержимого кучи  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::sum()// Суммаа элементов  {  int sum = 0;  std::cout << "Содержимое кучи:" << std::endl;  for (int i = 0; i < size; ++i)  {  AAA\* item = (AAA\*)storage[i];  sum += item->x;  }  std::cout << sum << std::endl;  }  // Извлечение минимального элемента из кучи  void\* Heap::extractMin()  {  void\* rc = nullptr;  if (!isEmpty())  {  rc = storage[0];  storage[0] = storage[size - 1];  size--;  heapify(0);  }  return rc;  }  }  **main.cpp**  #include "Heap.h"  #include <iostream>  #include <Windows.h>  // Функция сравнения для элементов типа AAA  heap::CMP cmpAAA(void\* a1, void\* a2)  {  AAA\* A1 = (AAA\*)a1;  AAA\* 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;  }  int main()  {  // Установка кодовой страницы консоли для корректного отображения символов  SetConsoleOutputCP(1251);  SetConsoleCP(1251);  int k, choice, index;  // Создание кучи с максимальным размером 30 и функцией сравнения cmpAAA  heap::Heap h1 = heap::create(30, cmpAAA);  for (;;)  {  // Вывод меню для пользователя  std::cout << "1 - вывести содержимое кучи на экран" << std::endl;  std::cout << "2 - добавить элемент" << std::endl;  std::cout << "3 - удалить максимальный элемент" << std::endl;  std::cout << "4 - удалить минимальный элемент" << std::endl;  std::cout << "5 - удалить элемент по индексу" << std::endl;  std::cout << "6 - объединить две кучи" << std::endl;  std::cout << "7 - сумма всех элементов" << std::endl;  std::cout << "8 - удалить элемент по значению" << std::endl;  std::cout << "9 - очистить кучу" << std::endl;  std::cout << "0 - выход" << std::endl;  std::cout << "сделайте выбор" << std::endl;  std::cin >> choice;  switch (choice)  {  case 0:  exit(0);  case 1:  h1.scan(0);  break;  case 2:  {  // Добавление нового элемента в кучу  AAA\* a = new AAA;  std::cout << "введите ключ" << std::endl;  std::cin >> k;  a->x = k;  h1.insert(a);  }  break;  case 3:  h1.extractMax();  break;  case 4:  h1.extractMin();  break;  case 5:  std::cout << "введите индекс элемента для удаления" << std::endl;  std::cin >> index;  h1.extractI(index);  break;  case 6:  {  // Создание второй кучи и объединение с первой  heap::Heap h2 = heap::create(2, cmpAAA);  for (size\_t i = 0; i < 2; i++)  {  AAA\* a = new AAA;  std::cout << "введите ключ" << std::endl;  std::cin >> k;  a->x = k;  h2.insert(a);  }  h1.merge(h2);  }  break;  case 7:  h1.sum();// Суммаа элементов  break;  case 8:  std::cout << "введите значение элемента для удаления" << std::endl;  std::cin >> k;  h1.remove(new AAA{ k });  break;  case 9:  h1.clear();  break;  default:  std::cout << std::endl << "Введена неверная команда!" << std::endl;  }  }  return 0;  } |
| **Результат программы** |
|  |