Институт информационных технологий

Кафедра: Математическое и программное обеспечение ЭВМ

Дисциплина: Объектно-ориентированное программирование

ЛАБОРАТОРНАЯ РАБОТА № 2

Тема: Разработка абстрактных типов данных

Выполнил:

студент гр. 1ПИб-02-2оп-23

Кринкин Олег Алексеевич

Проверил:

ассистент Матевосян Ремик Артурович

ЗАДАНИЕ

1. Разработайте АТД – класс, обеспечивающий хранение объектов класса из ЛР1, согласно варианту задания (см. раздел прил. 1).

2. Интерфейс класса может содержать конструкторы и должен содержать деструктор.

3. Для описания элементов контейнера использовать вложенные структуры.

4. Интерфейс класса должен содержать функции:

a. Добавления;

b. Удаления;

c. Вывода содержимого;

d. Обработки (сортировка, поиск по условию);

e. Получения данных контейнера в виде массива.

Представлены данные в классе-контейнеры должны быть в виде динамического списка.

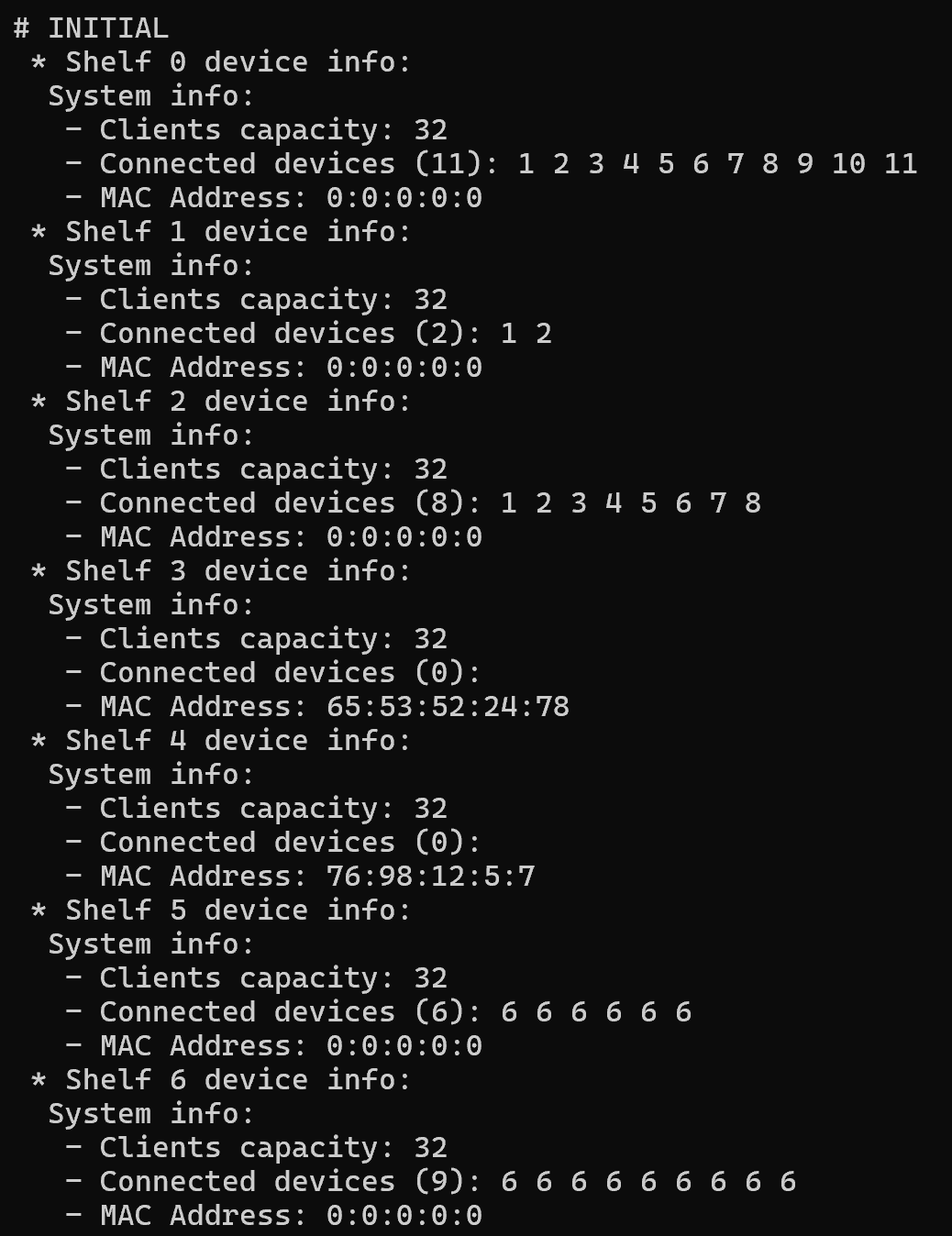
ХОД РАБОТЫ

1. Создан новый класс ServerRoom внутри которого создана структура – элемент динамического списка. Также создан указатель на первый и последний элемент.
2. Для класса определён конструктор по умолчанию, что инициализирует первый объект динамического списка. В коде объекты названы «полками» (Shelf). Также создан конструктор с параметром – данные что будет хранить «полка» - объект класса Switch.
3. Для класса определён деструктор, который проходит по всему динамическому списку и удаляет все его элементы.
4. Написана компонентная функция добавления, что создаёт в конце списка новый элемент и вносит в него указатель на объект Switch.
5. Написана компонентная функция удаления, что уничтожает последний элемент динамического списка, при этом не затрагивая указатель внутри него.
6. Для удобства работы написаны вспомогательные компонентные функции, возвращающие количество элементов в динамическом массиве, возвращающая элемент массива по индексу, возвращающая хранимое значение элемента по индексу.
7. Написана компонентная функция, что печатает на консоль информацию о элементах списка и вызывает функцию печати объекта Switch, хранимого в ней.
8. Создана функция поиска: функция проходит по всему динамическому списку и ищет объект Switch с указанным адресом.
9. Создана функция сортировки: Функция сортирует элементы динамического списка по количеству подключенных клиентов в объектах Switch. Сортировка происходит алгоритмом сортировки «пузырьком».
10. Написана функция получения элементов динамического списка в виде массива, которая внутри себя создаёт новый массив указателей на Switch, проходит по всему динамическому списку и добавляет хранимое значение каждого элемента в созданный внутри неё массив.
11. В основном файле (с функцией main()) создан набор объектов класса Switch и ServerRoom. У объекта ServerRoom вызваны различные компонентные функции для того, чтобы убедиться в его работоспособности. Результаты работы представлены в разделе ниже.

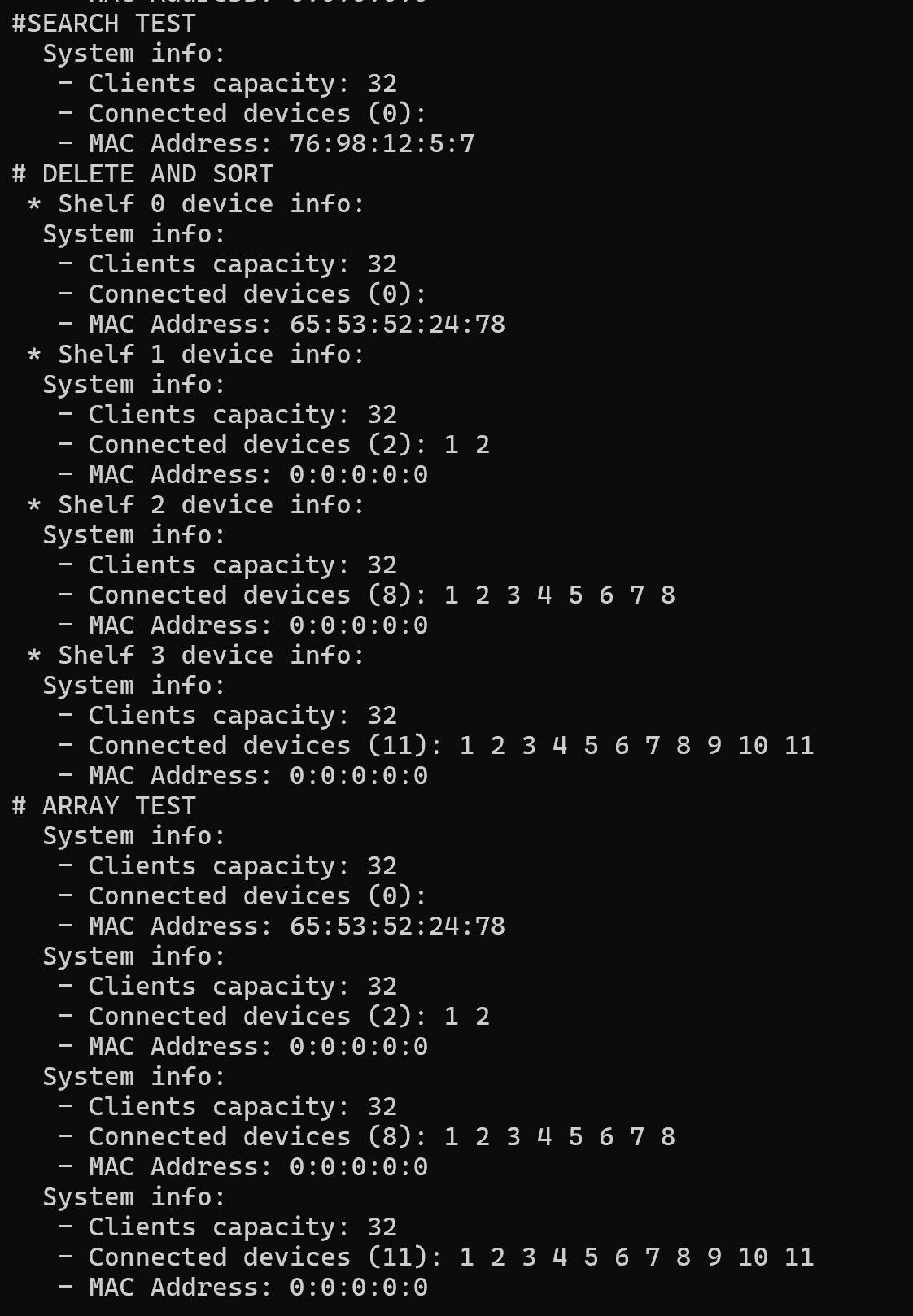
РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ РАБОТЫ

Результат исполнения программы

Изначальный список:



Результаты обработки:



Код программы:

|  |  |  |
| --- | --- | --- |
| net\_room.h | net\_room.cpp | Source.cpp |
| // Variant 5  #pragma once  #include "network.h"  class ServerRoom {  private:  // Структура-"полка" (динамический список)  struct Shelf {  const Switch\* device;  Shelf\* next;  };  // Указатели на первую и последнюю полку  Shelf\* first, \* last;  Shelf\* get\_shelf(int);  public:  ServerRoom();  ServerRoom(const Switch\*);  ~ServerRoom();  void add(const Switch\*);  void del();  void print(int);  void sort();  int get\_room\_size();  const Switch\* search(int\*);  const Switch\* get\_device(int);  const Switch\*\* get\_array();  }; | #include "net\_room.h"  #include <iostream>  ServerRoom::ServerRoom() {  /\*  \* Конструктор по умолчанию - создаёт пустую первую "полку".  \*/  this->first = new Shelf;  this->first->device = nullptr;  this->first->next = nullptr;  this->last = this->first;  }  ServerRoom::ServerRoom(const Switch\* device) {  /\*  \* Конструктор по умолчанию - создаёт первую "полку" с объектом Switch.  \*/  this->first = new Shelf;  this->first->device = device;  this->first->next = nullptr;  this->last = this->first;  }  ServerRoom::~ServerRoom() {  /\*  \* Декоструктор - удаляет все структуры-"полки", не затронув хранящиеся  \* в них объекты типа Switch.  \*/  Shelf\* current = this->first;  Shelf\* to\_delete = nullptr;  while (current->next != nullptr) {  to\_delete = current;  current = current->next;  delete to\_delete;  }  delete current;  }  ServerRoom::Shelf\* ServerRoom::get\_shelf(int shelf\_id) {  /\*  \* Функция возвращает указатель на "полку" под некоторым номером.  \* (приватна)  \*/  Shelf\* shelf = this->first;  for (int i = 0; i < shelf\_id; i++) {  if (shelf != nullptr)  shelf = shelf->next;  }  return shelf;  }  void ServerRoom::add(const Switch\* device) {  /\*  \* Функция добавляет "полку" в конец комнаты.  \*/  this->last->next = new Shelf;  this->last->next->device = device;  this->last = this->last->next;  this->last->next = nullptr;  }  void ServerRoom::del() {  /\*  \* Функция удаляет последнюю "полку".  \*/  Shelf\* prev = get\_shelf(get\_room\_size() - 1);  prev->next = nullptr;  delete this->last;  this->last = prev;  }  void ServerRoom::print(int shelf\_id) {  /\*  \* Функция печатает в консоли информацию о "полке" и её содержимом.  \*/  Shelf\* shelf = get\_shelf(shelf\_id);  if (shelf != nullptr && shelf->device != nullptr) {  std::cout << " \* Shelf " << shelf\_id << " device info:" << std::endl;  shelf->device->print\_info();  } else {  std::cout << " \* Shelf " << shelf\_id << " is empty!" << std::endl;  }  }  void ServerRoom::sort() {  /\*  \* Функция сортирует объекты Switch по количеству поключенных устройств  \* по возрастанию.  \*/  int room\_size = get\_room\_size();  for (int i = 0; i < room\_size; i++)  for (int j = 0; j < room\_size - 1 - i; j++) {  Shelf\* current = get\_shelf(j);  if (current->device->clients\_count() > current->next->device->clients\_count()) {  const Switch\* tmp;  tmp = current->next->device;  current->next->device = current->device;  current->device = tmp;  }  }  }  const Switch\* ServerRoom::search(int\* address) {  /\*  \* Функция поиска объекта Switch по адресу. Возвращает адрес объекта  \* или nullptr, если объект не найден.  \*/  Shelf\* current = this->first;  while (current->next != nullptr) {  for (int i = 0; i < 5; i++) {  if (current->device->get\_address()[i] != address[i]) {  current = current->next;  continue;  }  }  return current->device;  }  return nullptr;  }  int ServerRoom::get\_room\_size() {  /\*  \* Функция возвращает количество "полок" в комнате.  \*/  int count = 0;  Shelf\* current = this->first;  while (current->next != nullptr) {  if (current != nullptr)  current = current->next;  count++;  }  return count;  }  const Switch\* ServerRoom::get\_device(int shelf\_id) {  /\*  \* Функция возвращает указатель на объект Switch указанной "полки".  \*/  Shelf\* shelf = get\_shelf(shelf\_id);  return shelf->device;  }  const Switch\*\* ServerRoom::get\_array() {  /\*  \* Функция возвращает массив указателей на объекты Switch.  \*/  int size = get\_room\_size();  const Switch\*\* array = new const Switch\*[size];  for (int i = 0; i < size; i++) {  array[i] = get\_device(i);  }  return array;  } | #include "network.h"  #include "net\_room.h"  #include <iostream>  int main() {  Switch sw1 = Switch(new int[11]{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}, 11);  Switch sw2 = Switch(new int[2]{1, 2}, 2);  Switch sw3 = Switch(new int[8]{1, 2, 3, 4, 5, 6, 7, 8}, 8);  Switch sw4 = Switch(new int[5]{65, 53, 52, 24, 78});  Switch sw5 = Switch(new int[5]{76, 98, 12, 5, 7});  Switch sw6 = Switch(new int[6] {6, 6, 6, 6, 6, 6}, 6);  Switch sw7 = Switch(new int[9] {6, 6, 6, 6, 6, 6, 6, 6, 6}, 9);  Switch sw8 = Switch(new int[12] {6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6}, 12);  ServerRoom cab1 = ServerRoom(&sw1);  cab1.add(&sw2);  cab1.add(&sw3);  cab1.add(&sw4);  cab1.add(&sw5);  cab1.add(&sw6);  cab1.add(&sw7);  cab1.add(&sw8);  std::cout << "# INITIAL" << std::endl;  for (int i = 0; i < cab1.get\_room\_size(); i++)  cab1.print(i);  std::cout << "#SEARCH TEST" << std::endl;  cab1.search(new int[5]{76, 98, 12, 5, 7})->print\_info();  cab1.del();  cab1.del();  cab1.del();  cab1.sort();  std::cout << "# DELETE AND SORT" << std::endl;  for (int i = 0; i < cab1.get\_room\_size(); i++)  cab1.print(i);  std::cout << "# ARRAY TEST" << std::endl;  const Switch\*\* array = cab1.get\_array();  for (int i = 0; i < cab1.get\_room\_size(); i++)  array[i]->print\_info();  return 0;  } |

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были освоены навыки работы с несколькими объектами классов, их хранения и обработки внутри другого класса.