Институт информационных технологий

Кафедра: Математическое и программное обеспечение ЭВМ

Дисциплина: Объектно-ориентированное программирование

ЛАБОРАТОРНАЯ РАБОТА № 3

Тема: Перегрузка операторов

Выполнил:

студент гр. 1ПИб-02-2оп-23

Кринкин Олег Алексеевич

Проверил:

ассистент Матевосян Ремик Артурович

ЗАДАНИЕ

1. Ознакомьтесь с УМП по ООП часть 2 раздел VI.
2. Модифицируйте абстрактные типы данных, реализованные по заданию ЛР2, перегрузив минимум 3 ключевых оператора.
3. Проверьте работоспособность АТД на тестовом наборе данных.

* Обязательно перегрузите оператор вывода на консоль и один и операторов сравнения у АТД.
* Используйте эти операторы в классе-контейнере, в первую очереди в функции сортировки и вывода элементов АТД.

ХОД РАБОТЫ

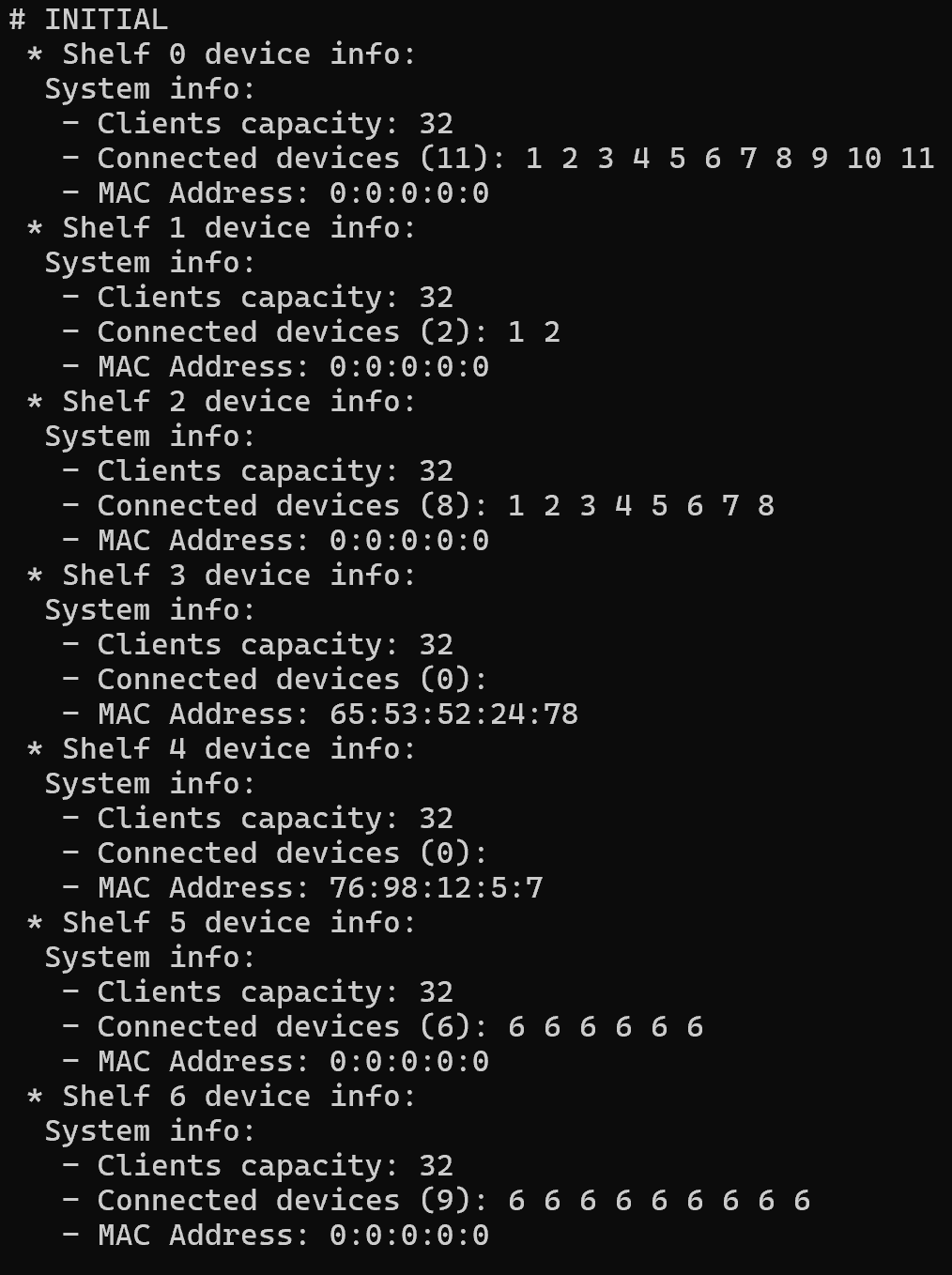
1. В классе ServerRoom объявлены операторы присваивания (+), декремента (--), взятия по индексу([]), сравнения (==), потокового вывода (<<).
2. Для определения оператора была использовано содержимое компонентной функции add(), которая добавляла в конец контейнера указанный объект Switch.
3. Для определения оператора декремента была использована компонентная функция del(), которая удаляла последний элемент в контейнере.
4. Для определения оператора взятия по индексу была использована функция get\_shelf(), которая возвращала указатель на объект Switch, находящийся в контейнере по некоторому индексу.
5. Для определения оператора потокового вывода была модифицирована компонентная функция print(), так чтобы она выводила информацию обо всех объектах в контейнере в определённый выходной поток. Сам оператор был определён при помощи дружественной функции, так как он является бинарным.
6. Определён оператор сравнения при помощи дружественной функции, которая сравнивает внутри себя значения вместительности и возвращает результат.
7. Все использования класса контейнера были изменены для соответствия новому варианту взаимодействия.
8. В класс Switch объявлен бинарный оператор больше (>), который впоследствии определён при помощи внешней дружественной функции, которая возвращает результат сравнения значения подключенных клиентов в объектах.
9. Добавленный в Switch оператор был использован в функции сортировки контейнерного класса.

РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ РАБОТЫ

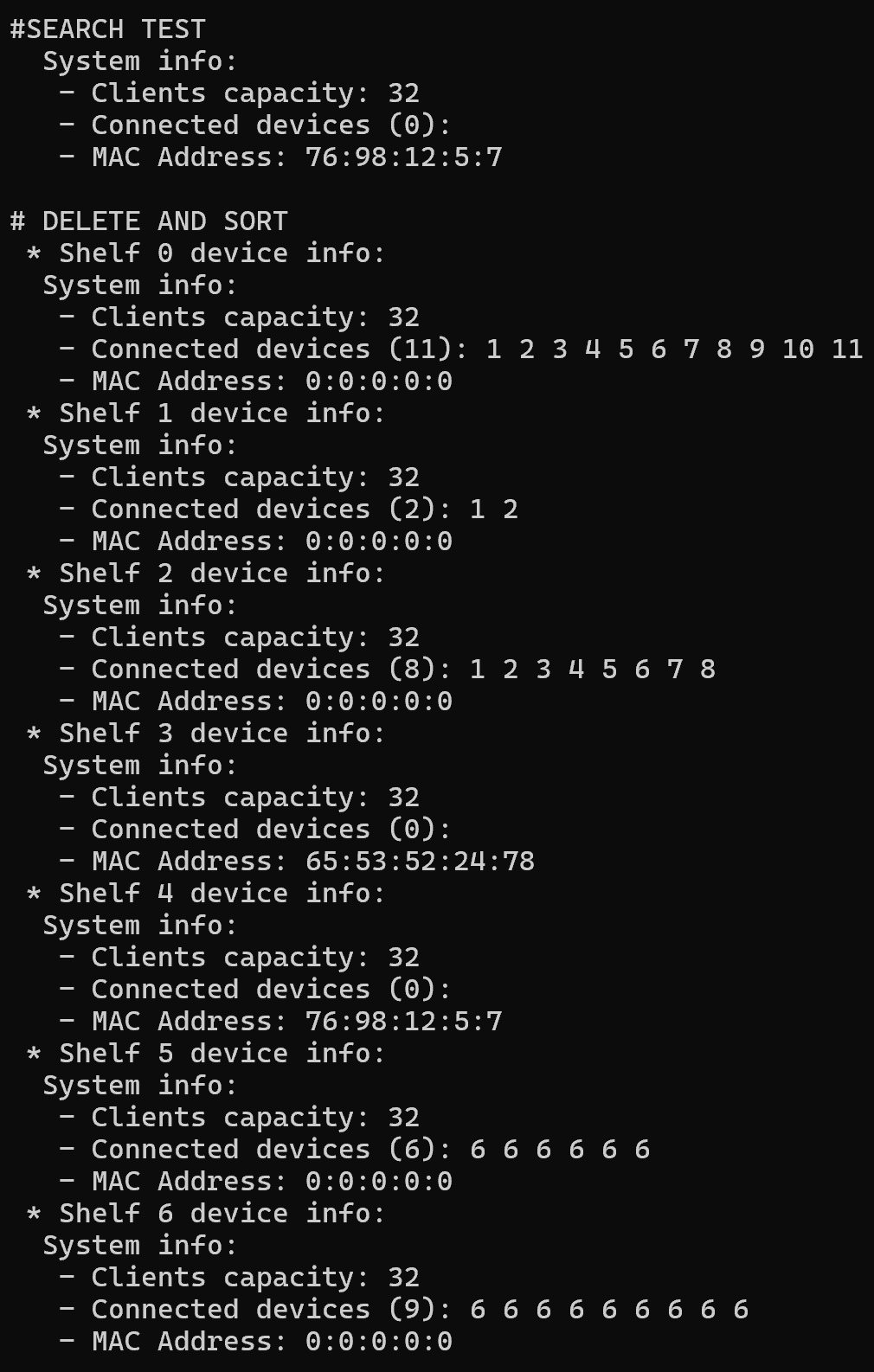
Результат исполнения программы

(Набор данных использован из предыдущей работы)

Изначальный список:



Результаты обработки:



Текст программы:

|  |  |  |
| --- | --- | --- |
| net\_room.h | net\_room.cpp | Source.cpp |
| // Variant 5  #pragma once  #include "network.h"  #include <ostream>  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(ServerRoom\*, int);  public:  ServerRoom();  ServerRoom(Switch&);  ~ServerRoom();  ServerRoom& operator+(Switch&); // Оператор присваивания ServerRoom и Switch  ServerRoom& operator--(int); // Оператор декремента  const Switch\* operator[](int); // Оператор взятия по индексу  friend bool operator==(ServerRoom&, ServerRoom&); // Оператор сравнения  friend std::ostream& operator<<(std::ostream& stream, ServerRoom&); // Оператор потокового вывода  void print(int);  void sort();  int get\_room\_size();  const Switch\* search(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(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(ServerRoom\* room, int shelf\_id) {  /\*  \* Функция возвращает указатель на "полку" под некоторым номером.  \* (приватна)  \*/  Shelf\* shelf = room->first;  for (int i = 0; i < shelf\_id; i++) {  if (shelf != nullptr)  shelf = shelf->next;  }  return shelf;  }  ServerRoom& ServerRoom::operator+(Switch& device) {  /\*  \* Оператор добавляет "полку" в конец комнаты.  \*/  this->last->next = new Shelf;  this->last->next->device = &device;  this->last = this->last->next;  this->last->next = nullptr;  return \*this;  }  ServerRoom& ServerRoom::operator--(int count) {  /\*  \* Оператор удаляет последнее количество "полок".  \*/  for (int i = 0; i < count; i++) {  Shelf\* prev = get\_shelf(this, get\_room\_size() - 1);  prev->next = nullptr;  delete this->last;  this->last = prev;  }  return \*this;  }  const Switch\* ServerRoom::operator[](int index) {  /\*  \* Оператор возвращает указатель на объект Switch указанной "полки".  \*/  Shelf\* shelf = get\_shelf(this, index);  return shelf->device;  }  bool operator==(ServerRoom& first, ServerRoom& second) {  /\*  \* Оператор проверяет равны ли две комнаты по размеру  \*/  return first.get\_room\_size() == second.get\_room\_size();  }  std::ostream& operator<<(std::ostream& stream, ServerRoom& room) {  /\*  \* Оператор печатает в консоль информацию обо всех устройствах в "комнате".  \*/  ServerRoom::Shelf\* shelf = room.first;  int size = room.get\_room\_size();  for (int i = 0; i < size; i++) {  ServerRoom::Shelf\* shelf = room.get\_shelf(&room, i);  if (shelf != nullptr && shelf->device != nullptr) {  stream << " \* Shelf " << i << " device info:" << std::endl;  shelf->device->print\_info();  } else  stream << " \* Shelf " << i << " is empty!" << std::endl;  shelf = shelf->next;  }  return stream;  }  void ServerRoom::print(int shelf\_id) {  /\*  \* Функция печатает в консоли информацию о "полке" и её содержимом.  \*/  Shelf\* shelf = get\_shelf(this, 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(this, j);  if (current->device > current->next->device) { // Использован оператор меньше  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\_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] = (\*this)[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 = cab1 + sw2;  cab1 = cab1 + sw3;  cab1 = cab1 + sw4;  cab1 = cab1 + sw5;  cab1 = cab1 + sw6;  cab1 = cab1 + sw7;  cab1 = cab1 + sw8;  std::cout << "# INITIAL" << std::endl;  std::cout << cab1 << std::endl; // Использование потокового вывода  std::cout << "#SEARCH TEST" << std::endl;  cab1.search(new int[5]{76, 98, 12, 5, 7})->print\_info();  std::cout << std::endl;  cab1--;  cab1--;  cab1--;  cab1.sort();  std::cout << "# DELETE AND SORT" << std::endl;  std::cout << cab1 << std::endl; // Использование потокового вывода  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 перегрузки операторов как унарных, так и бинарных, как компонентными функциями, так и внешними дружественными. Также объявленные для класса контейнера операторы были применены на практике при алгоритме сортировки и вывода на экран.