Институт информационных технологий

Кафедра: Математическое и программное обеспечение ЭВМ

Дисциплина: Объектно-ориентированное программирование

ЛАБОРАТОРНАЯ РАБОТА № 1

Тема: Разработка абстрактных типов данных

Выполнил:

студент гр. 1ПИб-02-2оп-23

Кринкин Олег Алексеевич

Проверил:

ассистент Матевосян Ремик Артурович

ЗАДАНИЕ

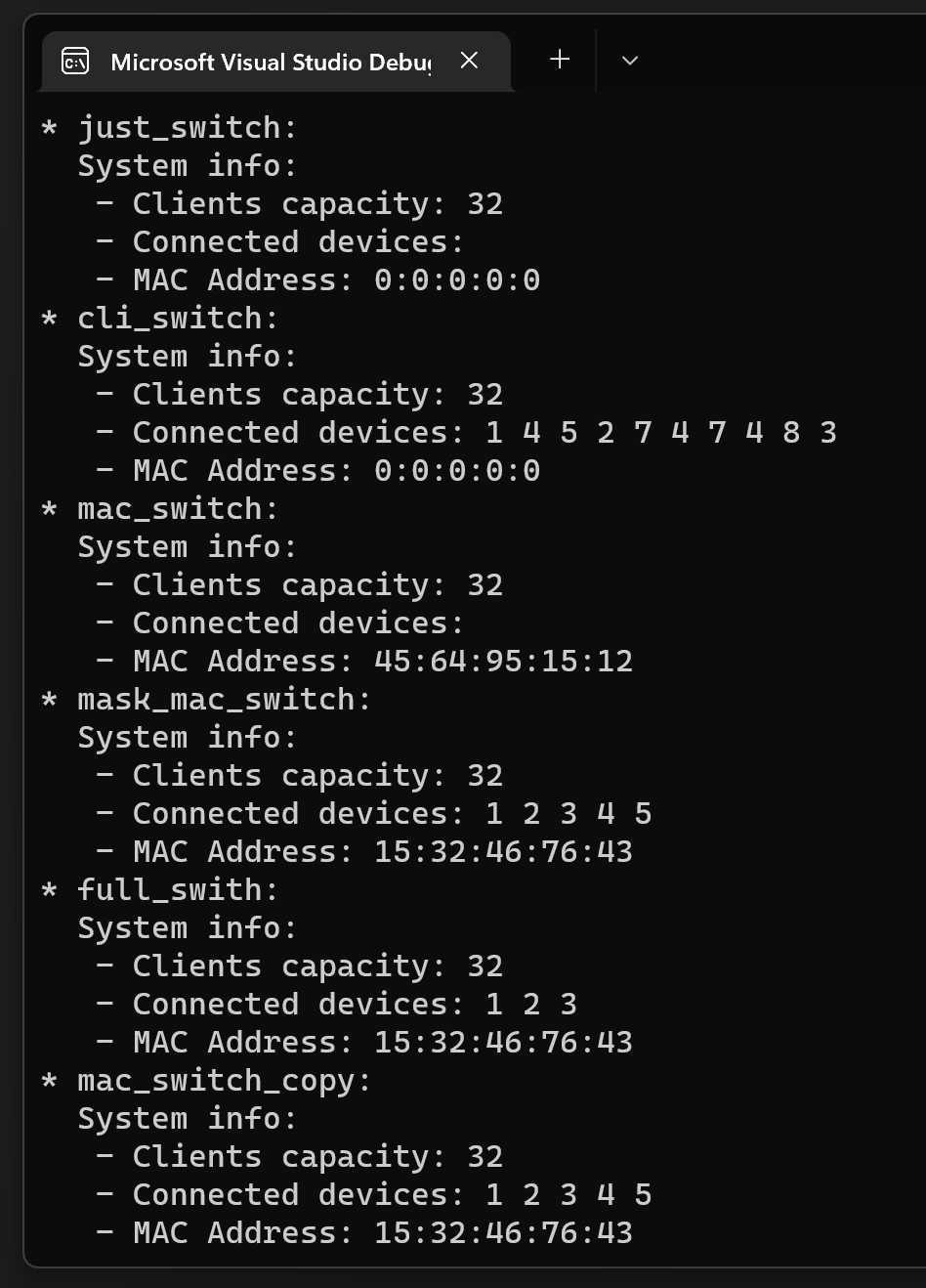
1. Ознакомьтесь с УМП по ООП часть 1 раздел IV.
2. Спроектируйте и разработайте алгоритм и программу, реализующую абстрактный тип данных (АТД) – класс, согласно варианту задания (см. прил. 1):
   1. Предусмотрите закрытую реализацию и открытый интерфейс.
   2. Интерфейс должен содержать функции вывода содержимого и функции доступа к некоторым полям.
   3. Элементы данных должны быть представлены из 5 полей, один из которых должен быть статическим, второй иметь тип указатель, а третий тип перечисление, остальные на свое усмотрение.
   4. Примените основные виды конструкторов и определитесь с деструктором класса.
   5. Проверьте работоспособность АТД на тестовом наборе данных.

ХОД РАБОТЫ

1. Создан новый класс Switch, который содержит следующий набор компонентных данных: целочисленный массив *packets* (передаваемые устройством пакеты данных), целочисленный масcив *clients* (идентификаторы подключенных к устройству клиентов), целочисленный массив *addres*s (MAC-адрес устройства), статическое константное целочисленное значение *cli\_cap* (возможное количество клиентов) и целочисленное значение *connected* (клиентов подключено).
2. Для класса определено несколько конструкторов: конструктор по умолчанию, конструктор с параметром clients, конструктор с параметром address, конструктор с полным набором параметров и копирующий конструктор. Конструктор по умолчанию внутри себя вызывает закрытую компонентную функцию set\_defaults(), которая устанавливает значения по умолчанию. Другие конструкторы также вызывают эту функцию, но затем устанавливают заданные для них значение при помощи функций сеттеров. Вынос всех действий сделан для соответствия принципу DRY.
3. Создана функция, которая печатает в консоль всю информацию об объекте, а также функция геттер, возвращающая MAC-адрес устройства.
4. В качестве уничтожаемых элементов в деструкторе выбраны поля *packets*, *clients*, *address*, т.к. они являются указателями.
5. В функции main() при помощи различных вариантов конструкторов с разным набором данных созданы несколько объектов разработанного класса Switch, после чего для каждого объекта была вызвана компонентная функция печати данных на экран.

РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ РАБОТЫ

Результат исполнения программы:



Код программы:

|  |  |  |
| --- | --- | --- |
| network.h | network.cpp | Source.cpp |
| // VARIANT A12  #pragma once  #include <iostream>  class Switch {  private:  static const int cli\_cap = 32; // Статическая  const double\* packets; // Указатель  int\* clients; // Массив  int\* address; // Произвольный  int connected; // Произвольный  void set\_defaults();  public:  Switch(); // Конструктор по умолчанию  Switch(const int\*, int); // clients  Switch(const int\*); // address  Switch(const int\*, int, const int\*); // mask, address  Switch(const double\*&, const int\*, int, const int\*); // Полный набор  Switch(const Switch&); // Копирующий  ~Switch();  void print\_info() const;  int\* get\_address() const;  void set\_address(const int\*);  void set\_clients(const int\*);  }; | // VARIANT A12  #define \_CRT\_SECURE\_NO\_WARNINGS  #include <iostream>  #include "network.h"  // Определение конструктора по умолчанию  Switch::Switch() {  set\_defaults();  }  // Определение конструктора с парамером clients  Switch::Switch(const int\* clients, int count) {  set\_defaults();  this->connected = count;  this->set\_clients(clients);  }  // Определение конструктора с парамером address  Switch::Switch(const int\* address) {  set\_defaults();  this->set\_address(address);  }  // Определение конструктора с парамерами clients и address  Switch::Switch(const int\* clients, int count, const int\* address) {  set\_defaults();  this->connected = count;  this->set\_clients(clients);  this->set\_address(address);  }  // Определение конструктора с полным набором параметров  Switch::Switch(const double\*& packets, const int\* clients, int count, const int\* address) {  this->set\_defaults();  this->connected = count;  this->packets = packets;  this->set\_clients(clients);  this->set\_address(address);  }  // Определение копирующего конструктора  Switch::Switch(const Switch& sw) {  this->packets = sw.packets;  this->connected = sw.connected;  this->set\_clients(sw.clients);  this->set\_address(sw.address);  };  // Деструктор  Switch::~Switch() {  delete this->packets;  delete[] this->clients;  delete[] this->address;  }  // Эта компонентная функция печатает данные из объекта на экран.  void Switch::print\_info() const {  std::cout << " System info: " << std::endl;  std::cout << " - Clients capacity: " << this->cli\_cap << std::endl;  std::cout << " - Connected devices: ";  for (int i = 0; i < this->connected; i++)  std::cout << this->clients[i] << " ";  std::cout << std::endl;  std::cout << " - MAC Address: ";  for (int i = 0; i < 5; i++)  std::cout << this->address[i] << ":";  std::cout << "\b" << " " << std::endl;  }  // Эта компонентная функция возвращает значение поля address (геттер).  int\* Switch::get\_address() const {  return this->address;  };  // Эта компонентная функция устанавливает стандатные значения для объекта (сеттер).  void Switch::set\_defaults() {  this->connected = 0;  this->packets = nullptr;  this->clients = new int[this->cli\_cap]();  this->address = new int[5]{0, 0, 0, 0, 0};  }  // Эта компонентная функция устанавливает значение поля address (сеттер).  void Switch::set\_address(const int\* address) {  const int addr\_len = 5;  this->address = new int[addr\_len];  for (int i = 0; i < addr\_len; i++)  this->address[i] = address[i];  }  // Эта компонентная функция устанавливает значение поля clients.  void Switch::set\_clients(const int\* clients) {  this->clients = new int[this->cli\_cap]();  for (int i = 0; i < this->connected; i++)  this->clients[i] = clients[i];  } | #include <iostream>  #include "network.h"  int main() {  const double\* packet = new double[10] {  143454.33, 657.78, 78.98, 45.87, 633.767,  6789.7889, 45.876, 345.8, 887.90, 77.89  }; // Некоторый передаваемый пакет данных  Switch just\_switch; // Объект с конструктором по умолчанию  Switch cli\_switch = Switch(new int[10] {1, 4, 5, 2, 7, 4, 7, 4, 8, 3}, 10); // Объект с конструктором clients  Switch mac\_switch = Switch(new int[5]{45, 64, 95, 15, 12}); // Объект с конструктором address  Switch mask\_mac\_switch = Switch(new int[5] {1, 2, 3, 4, 5}, 5, new int[5]{15, 32, 46, 76, 43}); // Объект с конструктором clients, address  Switch full\_swith = Switch(packet, new int[3] {1, 2, 3}, 3, new int[5] {15, 32, 46, 76, 43}); // Объект с полным конструктором  Switch mask\_mac\_switch\_copy = Switch(mask\_mac\_switch); // Объект с копирующим конструктором  std::cout << "\* just\_switch: " << std::endl;  just\_switch.print\_info();  std::cout << "\* cli\_switch: " << std::endl;  cli\_switch.print\_info();  std::cout << "\* mac\_switch: " << std::endl;  mac\_switch.print\_info();  std::cout << "\* mask\_mac\_switch: " << std::endl;  mask\_mac\_switch.print\_info();  std::cout << "\* full\_swith: " << std::endl;  full\_swith.print\_info();  std::cout << "\* mac\_switch\_copy: " << std::endl;  mask\_mac\_switch\_copy.print\_info();    return 0;  } |

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были изучены принципы и механизмы создания абстрактных типов данных, правила доступа к элементам, изучен синтаксис и семантика определения и вызова конструкторов и деструкторов, а также приобретены практические навыки создания и уничтожения объектов, изучены особенности применения различных видов конструкторов.