|  |  |
| --- | --- |
| Картина, която съдържа скица, символ, мотив, Симетрия  Описанието е генерирано автоматично | **ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ**  **ФАКУЛТЕТ ПО ИНДУСТРИАЛНИ ТЕХНОЛОГИИ** |

**КУРСОВ ПРОЕКТ**

**Дисциплина: ,,Обектно ориентирано програмиране‘‘**

**Тема: 230113**

**Изготвил: Антон Антонов Русев**

**Факултетен номер: 361224031**

**Група: 21А**

**Специалност: ИСИИ , Курс: 1**

**София, 2024**

**Текст на заданието**

**Да се проектира и реализира приложение, което служи за обработка на информация и за управление на системи в превозни средства.**

* 1. Декларирайте клас RadioStation с атрибути за име, честота в MHz, качество в % и текущо състояние (visible/grayed\_out/not\_visible) с private достъп.
     1. Добавете конструктори, деструктор, методи за достъп, метод за отпечатване на информация.
     2. Добавете метод за определяне на текущото състояние на радио-станция според следните условия:

При качество >45% - станцията е в състояние visible;

При качество в интервала (35%, 45%) - - станцията е в състояние grayed\_out;

При качество <35% - станцията е в състояние not\_visible;

* 1. Декларирайте функция main(), в която се обработва информация за радио станции .
     1. Въведете от клавиатурата данни за 6 обекта от клас RadioStation като ги съхранявате в масив.
     2. Отпечатайте на екрана информацията за всички радио станции.
     3. Отпечатайте на екрана имената и честотите на радио станциите със състояние visible.
     4. Реализирайте функция за търсене на радио станция по зададена честота.
     5. Реализирайте функция за търсене на радио станция по име

**Блоков алгоритъм на разработеното програмно осигуряване**

**+-----------------------------------+**

**| Начало на програмата |**

**+-----------------------------------+**

**|**

**v**

**+-----------------------------------+**

**| Създаване на обекти RadioStation |**

**+-----------------------------------+**

**|**

**v**

**+-----------------------------------+**

**| Отпечатване на всички станции |**

**+-----------------------------------+**

**|**

**v**

**+-----------------------------------+**

**| Отпечатване на "visible" станции |**

**+-----------------------------------+**

**|**

**v**

**+-----------------------------------+**

**| Показване на меню за търсене |**

**+-----------------------------------+**

**|**

**v**

**+-----------------+-----------------+**

**| Търсене по честота | Търсене по име |**

**+-----------------+-----------------+**

**| |**

**v v**

**+----------------+ +----------------+**

**| Търсене по честота | | Търсене по име |**

**+----------------+ +----------------+**

**|**

**v**

**+-----------------------------------+**

**| Край на програмата |**

**+-----------------------------------+**

**Описание на използваните модули (класове) - елементи и предназначение**

**Клас RadioStation**

**Цел**:  
Класът **RadioStation** е основният модул на програмата. Той съхранява информацията за всяка радиостанция, като името ѝ, честотата, качеството и състоянието. Също така, класът предлага различни методи, които ни помагат да обработваме тези данни.

**Атрибути (Променливи):**

* **string name**  
  *Т*ова е променлива от тип string, която съдържа името на радиостанцията. Тя ни помага да знаем как се казва всяка радиостанция.
* **float frequency**  
  Това е променлива от тип float, която съхранява честотата на радиостанцията в MHz. Чрез нея можем да разберем на коя честота работи радиостанцията.
* **int quality**  
  Това е променлива от тип int, която показва качеството на сигнала на радиостанцията в проценти.
* **string status**  
  Това е променлива от тип string, която съдържа състоянието на радиостанцията.

**Методите (Функции):**

* **RadioStation(const string& n, float f, int q, const string& s)**  
  Това е конструктор. Когато създаваме нова радиостанция, този конструктор ни позволява да зададем името, честотата, качеството и състоянието.  
  Този метод създава нова радиостанция с дадените параметри.
* **string getName() const**  
  Метод, който връща името на радиостанцията.  
  Когато извикаме този метод, той ще ни даде името на радиостанцията.
* **float getFrequency() const**  
  Метод, който връща честотата на радиостанцията.  
  Ще ни даде честотата.
* **int getQuality() const**  
  Метод, който връща качеството на сигнала на радиостанцията.  
  Ще ни даде процента на качеството.
* **string getStatus() const**  
  Метод, който връща състоянието на радиостанцията.  
  Този метод ни казва дали радиостанцията е в състояние "visible", "grayed\_out" или "not\_visible".
* **void setStatus(const string& newStatus)**  
  Метод, който позволява да променим състоянието на радиостанцията.  
  С него можем да зададем ново състояние на радиостанцията, ако това е необходимо.
* **void updateStatus()**  
  Метод, който актуализира състоянието на радиостанцията въз основа на качеството на сигнала.  
  Ако качеството на сигнала е по-голямо от 45%, състоянието ще бъде "visible". Ако е между 35% и 45%, ще бъде "grayed\_out", а ако е под 35%, ще бъде "not\_visible".
* **void printInfo() const**  
  Метод, който отпечатва на екрана информация за радиостанцията.  
  Показва данни за радиостанцията.
* **static void searchByFrequency(const RadioStation stations[], int size, float targetFrequency)**  
  Статичен метод, който търси радиостанция по дадена честота.  
  Чрез този метод можем да потърсим радиостанция, която работи на определена честота и ако я намерим, ще отпечата информация за нея.
* **static void searchByName(const RadioStation stations[], int size, const string& targetName)**  
   Статичен метод, който търси радиостанция по име.  
   С този метод можем да търсим радиостанция по име и ако я намерим, ще отпечата информация за нея.

**Променливи:**

* **RadioStation stations[6]**  
  Това е масив, който съдържа 6 обекта от клас **RadioStation**.  
  Този масив съхранява информацията за всички радиостанции, които ще използваме в програмата.
* **int choice**  
  Това е променлива, която пази избора на потребителя.  
  Потребителят избира какво да търси — честота или име.
* **float targetFrequency**  
  *Какво е това?* Това е променлива, която съхранява честотата, по която потребителят иска да търси радиостанция.  
  *Какво прави?* Тази променлива ни позволява да въведем честотата, по която искаме да търсим.
* **string targetName**  
  Това е променлива, която съхранява името на радиостанцията, по което потребителят иска да търси.  
  Тази променлива ни позволява да въведем името на радиостанцията, по което искаме да търсим.

**Функции:**

* **main()**  
  Това е основната функция, в която се изпълнява цялата програма.

 В нея се създават радиостанциите, отпечатват се техните данни, предлага се избор за търсене по честота или по име и след това се извършва съответното търсене.

**Общо описание за функционирането на програмата (вход/изход)**

**Вход (Input):**

1. **Данни за радиостанции**  
   Програмата започва с това, че потребителят трябва да въведе данни за 6 радиостанции. За всяка радиостанция се въвеждат:
   * Име на радиостанцията.
   * Честота на радиостанцията.
   * Качество на сигнала.

Тези данни се съхраняват в масив от обекти на класа **RadioStation**.

1. **Избор за търсене**  
   След като въведем данни за радиостанциите, потребителят има възможност да избере:
   * Търсене по **честота**: Потребителят въвежда конкретна честота, по която да се търси радиостанция.
   * Търсене по **име**: Потребителят въвежда име на радиостанция и програмата ще извърши търсене по това име.

**Изход (Output):**

1. **Информация за всички радиостанции** – Програмата показва детайлна информация за всяка радиостанция, включително името, честотата, качеството и състоянието (visible, grayed\_out или not\_visible).
2. **Радиостанции със състояние visible** – В края програмата ще отпечата само тези радиостанции, които са с състояние "visible".
3. **Резултати от търсене по честота или име** – Потребителят може да търси радиостанции по честота или по име и програмата ще отпечата съответната информация, ако такава радиостанция съществува.

**Листинг на source (изходния) код на програмата**

#include <iostream>

#include <string>

using namespace std;

class RadioStation {

private:

string name;

float frequency;

int quality;

string status;

public:

RadioStation(const string& n, float f, int q, const string& s)

: name(n), frequency(f), quality(q), status(s) {}

string getName() const {

return name;

}

float getFrequency() const {

return frequency;

}

int getQuality() const {

return quality;

}

string getStatus() const {

return status;

}

void setStatus(const string& newStatus) {

status = newStatus;

}

void updateStatus() {

if (quality > 45) {

status = "visible";

} else if (quality >= 35 && quality <= 45) {

status = "grayed\_out";

} else {

status = "not\_visible";

}

}

void printInfo() const {

cout << "Radio Station Info:\n";

cout << "Name: " << name << "\n";

cout << "Frequency: " << frequency << " MHz\n";

cout << "Quality: " << quality << "%\n";

cout << "Status: " << status << "\n";

}

void inputData() {

updateStatus();

}

static void searchByFrequency(const RadioStation stations[], int size, float targetFrequency) {

bool found = false;

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

if (stations[i].getFrequency() == targetFrequency) {

cout << "Found Radio Station with Frequency " << targetFrequency << " MHz:\n";

stations[i].printInfo();

found = true;

break;

}

}

if (!found) {

cout << "No radio station found with Frequency " << targetFrequency << " MHz.\n";

}

}

static void searchByName(const RadioStation stations[], int size, const string& targetName) {

bool found = false;

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

if (stations[i].getName() == targetName) {

cout << "Found Radio Station with Name '" << targetName << "':\n";

stations[i].printInfo();

found = true;

break;

}

}

if (!found) {

cout << "No radio station found with Name '" << targetName << "'.\n";

}

}

};

int main() {

RadioStation stations[6] = {

RadioStation("Rock 1", 95.5, 60, "visible"),

RadioStation("Jazz FM", 101.5, 40, "grayed\_out"),

RadioStation("Classical Music", 88.0, 25, "not\_visible"),

RadioStation("Vitosha", 104.3, 45, "grayed\_out"),

RadioStation("BBC News", 100.0, 70, "visible"),

RadioStation("The Voice", 91.2, 33, "not\_visible")

};

cout << "\nAll Radio Stations:\n";

for (int i = 0; i < 6; ++i) {

stations[i].printInfo();

cout << "\n";

}

cout << "\nRadio stations with 'visible' status:\n";

for (int i = 0; i < 6; ++i) {

if (stations[i].getStatus() == "visible") {

cout << "Name: " << stations[i].getName() << ", Frequency: " << stations[i].getFrequency() << " MHz\n";

}

}

int choice;

cout << "\nSearch for a radio station by:\n";

cout << "1. Frequency\n";

cout << "2. Name\n";

cout << "Enter your choice (1 or 2): ";

cin >> choice;

if (choice == 1) {

float targetFrequency;

cout << "Enter frequency to search for: ";

cin >> targetFrequency;

RadioStation::searchByFrequency(stations, 6, targetFrequency);

} else if (choice == 2) {

string targetName;

cout << "Enter name to search for: ";

cin.ignore();

getline(cin, targetName);

RadioStation::searchByName(stations, 6, targetName);

} else {

cout << "Invalid choice. Please select 1 or 2.\n";

}

return 0;

}

**Резултати от изпълнението на програмата**

All Radio Stations:

Radio Station Info:

Name: Rock 1

Frequency: 95.5 MHz

Quality: 60%

Status: visible

Radio Station Info:

Name: Jazz FM

Frequency: 101.5 MHz

Quality: 40%

Status: grayed\_out

Radio Station Info:

Name: Classical Music

Frequency: 88 MHz

Quality: 25%

Status: not\_visible

Radio Station Info:

Name: Vitosha

Frequency: 104.3 MHz

Quality: 45%

Status: grayed\_out

Radio Station Info:

Name: BBC News

Frequency: 100 MHz

Quality: 70%

Status: visible

Radio Station Info:

Name: The Voice

Frequency: 91.2 MHz

Quality: 33%

Status: not\_visible

Radio stations with 'visible' status:

Name: Rock 1, Frequency: 95.5 MHz

Name: BBC News, Frequency: 100 MHz

Search for a radio station by:

1. Frequency

2. Name

Enter your choice (1 or 2): 1

Enter frequency to search for: 100.0

Found Radio Station with Frequency 100 MHz:

Radio Station Info:

Name: BBC News

Frequency: 100 MHz

Quality: 70%

Status: visible