Исполнитель Казаков Д.И.

Группа ИКТС-32

Вариант работы: 9

**РЕЗУЛЬТАТ РАБОТЫ**

**Функция “Чтение данных из файла в динамический массив”**

| RadioTechnics\* loadFromFile(int& count) {  RadioTechnics\* records = nullptr;  string filename;   while (true) {  cout << "Enter the filename to load data: ";  cin >> filename;   ifstream file(filename);  if (!file) {  cerr << "Error opening file. Please try again.\n";  continue;  }   file >> count;  if (count <= 0) {  cerr << "Invalid record count in the file. Please try again.\n";  file.close();  continue;  }   records = new RadioTechnics[count];  for (int i = 0; i < count; ++i) {  file >> records[i].id >> records[i].name >> records[i].manufacturer  >> records[i].purpose >> records[i].frequency >> records[i].power  >> records[i].weight;  }  file.close();  cout << "Data successfully loaded from file: " << filename << "\n";  break;  }  return records; } |
| --- |

**Функция “Просмотр данных, хранящихся в массиве”**

| void displayRecords(const RadioTechnics\* records, int count) {  if (count == 0) {  cout << "No data available.\n";  return;  }   cout << left << setw(5) << "ID" << setw(15) << "Name" << setw(20) << "Manufacturer"  << setw(15) << "Purpose" << setw(10) << "Frequency" << setw(10) << "Power"  << setw(10) << "Weight\n";  cout << string(75, '-') << "\n";   for (int i = 0; i < count; ++i) {  cout << setw(5) << records[i].id << setw(15) << records[i].name << setw(20)  << records[i].manufacturer << setw(15) << records[i].purpose  << setw(10) << records[i].frequency << setw(10) << records[i].power  << setw(10) << records[i].weight << "\n";  } } |
| --- |

**Функция “Корректировка данных заданной строки”**

| void editRecord(RadioTechnics\* records, int count) {  int id = getValidatedInt("Enter the ID of the record to edit: ");  bool found = false;   for (int i = 0; i < count; ++i) {  if (records[i].id == id) {  found = true;  cout << "Editing record with ID " << id << ":\n";  cout << "New name: ";  cin >> records[i].name;  cout << "New manufacturer: ";  cin >> records[i].manufacturer;  cout << "New purpose: ";  cin >> records[i].purpose;  records[i].frequency = getValidatedFloat("New frequency (Hz): ");  records[i].power = getValidatedDouble("New power (W): ");  records[i].weight = getValidatedDouble("New weight (kg): ");  cout << "Record updated successfully.\n";  break;  }  }   if (!found) {  cout << "Record with ID " << id << " not found.\n";  } } |
| --- |

**Функция “Вычисления”**

| void calculateAverageWeight(const RadioTechnics\* records, int count) {  if (count == 0) {  cout << "No data available.\n";  return;  }   double totalWeight = 0;  for (int i = 0; i < count; ++i) {  totalWeight += records[i].weight;  }   cout << "Average weight: " << totalWeight / count << " kg\n"; } |
| --- |

**Функция “Сохранение данных в новом файле”**

| void saveToFile(const RadioTechnics\* records, int count, const string& filename) {  ofstream file(filename);  if (!file) {  cerr << "Error opening file for writing.\n";  return;  }   file << count << "\n";  for (int i = 0; i < count; ++i) {  file << records[i].id << " " << records[i].name << " " << records[i].manufacturer << " "  << records[i].purpose << " " << records[i].frequency << " " << records[i].power  << " " << records[i].weight << "\n";  }  file.close();  cout << "Data successfully saved to file " << filename << ".\n"; } |
| --- |

**Файл main.cpp**

| #include <iostream> #include <fstream> #include <iomanip> #include <string> #include <limits>  using namespace std;  struct RadioTechnics {  int id;  string name;  string manufacturer;  string purpose;  float frequency; // Hz  double power; // W  double weight; // kg };  RadioTechnics\* loadFromFile(int& count); void displayRecords(const RadioTechnics\* records, int count); void editRecord(RadioTechnics\* records, int count); void deleteRecord(RadioTechnics\*& records, int& count); void calculateAverageWeight(const RadioTechnics\* records, int count); void saveToFile(const RadioTechnics\* records, int count, const string& filename); int getValidatedInt(const string& prompt); double getValidatedDouble(const string& prompt); float getValidatedFloat(const string& prompt);  int main() {  RadioTechnics\* records = nullptr;  int count = 0;   records = loadFromFile(count);   if (!records) {  cout << "Failed to load data. Exiting program.\n";  return 1;  }   int choice;  do {  cout << "\nProgram Menu:\n";  cout << "1. View data\n";  cout << "2. Edit record\n";  cout << "3. Delete record\n";  cout << "4. Calculate average weight\n";  cout << "5. Save data to a file\n";  cout << "0. Exit\n";  cout << "Enter your choice: ";  choice = getValidatedInt("");   switch (choice) {  case 1:  displayRecords(records, count);  break;  case 2:  editRecord(records, count);  break;  case 3:  deleteRecord(records, count);  break;  case 4:  calculateAverageWeight(records, count);  break;  case 5: {  string filename;  cout << "Enter the filename to save data: ";  cin >> filename;  saveToFile(records, count, filename);  break;  }  case 0:  cout << "Exiting program.\n";  break;  default:  cout << "Invalid choice. Try again.\n";  }  } while (choice != 0);   delete[] records;  return 0; }   RadioTechnics\* loadFromFile(int& count) {  RadioTechnics\* records = nullptr;  string filename;   while (true) {  cout << "Enter the filename to load data: ";  cin >> filename;   ifstream file(filename);  if (!file) {  cerr << "Error opening file. Please try again.\n";  continue;  }   file >> count;  if (count <= 0) {  cerr << "Invalid record count in the file. Please try again.\n";  file.close();  continue;  }   records = new RadioTechnics[count];  for (int i = 0; i < count; ++i) {  file >> records[i].id >> records[i].name >> records[i].manufacturer  >> records[i].purpose >> records[i].frequency >> records[i].power  >> records[i].weight;  }  file.close();  cout << "Data successfully loaded from file: " << filename << "\n";  break;  }  return records; }  void displayRecords(const RadioTechnics\* records, int count) {  if (count == 0) {  cout << "No data available.\n";  return;  }   cout << left << setw(5) << "ID" << setw(15) << "Name" << setw(20) << "Manufacturer"  << setw(15) << "Purpose" << setw(10) << "Frequency" << setw(10) << "Power"  << setw(10) << "Weight\n";  cout << string(75, '-') << "\n";   for (int i = 0; i < count; ++i) {  cout << setw(5) << records[i].id << setw(15) << records[i].name << setw(20)  << records[i].manufacturer << setw(15) << records[i].purpose  << setw(10) << records[i].frequency << setw(10) << records[i].power  << setw(10) << records[i].weight << "\n";  } }  void editRecord(RadioTechnics\* records, int count) {  int id = getValidatedInt("Enter the ID of the record to edit: ");  bool found = false;   for (int i = 0; i < count; ++i) {  if (records[i].id == id) {  found = true;  cout << "Editing record with ID " << id << ":\n";  cout << "New name: ";  cin >> records[i].name;  cout << "New manufacturer: ";  cin >> records[i].manufacturer;  cout << "New purpose: ";  cin >> records[i].purpose;  records[i].frequency = getValidatedFloat("New frequency (Hz): ");  records[i].power = getValidatedDouble("New power (W): ");  records[i].weight = getValidatedDouble("New weight (kg): ");  cout << "Record updated successfully.\n";  break;  }  }   if (!found) {  cout << "Record with ID " << id << " not found.\n";  } }  void deleteRecord(RadioTechnics\*& records, int& count) {  int id = getValidatedInt("Enter the ID of the record to delete: ");  int index = -1;   for (int i = 0; i < count; ++i) {  if (records[i].id == id) {  index = i;  break;  }  }   if (index == -1) {  cout << "Record with ID " << id << " not found.\n";  return;  }   for (int i = index; i < count - 1; ++i) {  records[i] = records[i + 1];  }  --count;   RadioTechnics\* temp = new RadioTechnics[count];  for (int i = 0; i < count; ++i) {  temp[i] = records[i];  }  delete[] records;  records = temp;   cout << "Record with ID " << id << " deleted.\n"; }  void calculateAverageWeight(const RadioTechnics\* records, int count) {  if (count == 0) {  cout << "No data available.\n";  return;  }   double totalWeight = 0;  for (int i = 0; i < count; ++i) {  totalWeight += records[i].weight;  }   cout << "Average weight: " << totalWeight / count << " kg\n"; }  void saveToFile(const RadioTechnics\* records, int count, const string& filename) {  ofstream file(filename);  if (!file) {  cerr << "Error opening file for writing.\n";  return;  }   file << count << "\n";  for (int i = 0; i < count; ++i) {  file << records[i].id << " " << records[i].name << " " << records[i].manufacturer << " "  << records[i].purpose << " " << records[i].frequency << " " << records[i].power  << " " << records[i].weight << "\n";  }  file.close();  cout << "Data successfully saved to file " << filename << ".\n"; }   int getValidatedInt(const string& prompt) {  int value;  while (true) {  cout << prompt;  cin >> value;  if (!cin.fail()) {  return value;  }  cin.clear();  cin.ignore(numeric\_limits<streamsize>::max(), '\n');  cout << "Invalid input. Try again.\n";  } }  double getValidatedDouble(const string& prompt) {  double value;  while (true) {  cout << prompt;  cin >> value;  if (!cin.fail()) {  return value;  }  cin.clear();  cin.ignore(numeric\_limits<streamsize>::max(), '\n');  cout << "Invalid input. Try again.\n";  } }  float getValidatedFloat(const string& prompt) {  float value;  while (true) {  cout << prompt;  cin >> value;  if (!cin.fail()) {  return value;  }  cin.clear();  cin.ignore(numeric\_limits<streamsize>::max(), '\n');  cout << "Invalid input. Try again.\n";  } } |
| --- |

**ЗАКЛЮЧЕНИЕ**

*В ходе этой работы я изучил структуры языка C++ и получил практические навыки программирования С++. все гут)))*