Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа 4

По дисциплине «Основы алгоритмизации программирования»

На тему «Обработка символьной информации»

Выполнил:

Студент 1 курса 6 группы

Кучерук Николай Петрович

Преподаватель: асс. Андронова М.В.

2023, Минск

**Задание**

| **№ варианта** | **Условие задачи** |
| --- | --- |
| **9** | 1. Написать программу, реализующую вставку подстроки **St** длиной **n1** в строку **S** с позиции номер **n2**.  2. Дана строка, состоящая из групп нулей и единиц. Подсчитать количество символов в самой длинной группе. |

|  |
| --- |
| **Код программы** |
| #include <iostream>  #include <fstream>  #include <string>  using namespace std;  struct student//создаем структуру,в которую добавляем все параметры,которые хотим знать о студентах  {  string LastName;  string FirstName;  string FatherName;  string date;  string major;  string group;  string Fac;  string gpa;  }stud;  student student\_list[30];//массив,позволяющий хранить данные о группе студентов в рамках одной переменной  int currentsize();//данная функция нужна для нахождения кол-ва уже записаных студентов в файле до начала программы  void input();  void output();  void del();  void fill();//данная функция нужна,чтобы заполнить наш массив данными из файла  void find\_lastname(string);  int current\_size = currentsize(), size = 30;  int main()  {  system("color 70");  setlocale(LC\_ALL, "Russian");  int choice, count = 1; string fio;  do  {  cout << "\n1.Ввод данных с клавиатуры и запись в файл\n";  cout << "2.Вывод данных из файла\n";  cout << "3.Поиск по фамилии\n";  cout << "4.Удалить строку \n";  cout << "0.Выход из программы\n\n";  cout << "Введите номер операции: ";  cin >> choice;  switch (choice)  {  case 1: cout << "Введите количество студентов: ";  input(); break;  case 2: output(); break;  case 3: { cout << "Введите фамилию: ";  cin >> fio;  find\_lastname(fio); break;  }  case 4: del(); break;  case 0: exit(0); break;  }  } while (choice != 0);  }  void input() {  int number;  char buffer[200]; string buf;  ofstream file("database.txt",ios::app);//открываем файл для записи,не удаляя содержимое  if (file.fail())  {  cout << "\n Ошибка открытия файла";  exit(1);  }  cout << "Введите кол-во студентов,которые вы хотите добавить в реестр, на данный момент записано " << current\_size << " (макс.30): "; cin >> number;  if (current\_size + number >= 30) { cout << "Превышен лимит студентов."; exit(1); }  for (int p = current\_size; p < current\_size+number; p++)//записываем значения параметров структуры и записываем их в файл  {  cout << endl << "Фамилия: "; cin >> student\_list[p].LastName; file << student\_list[p].LastName << ' ';  cout << "Имя: "; cin >> student\_list[p].FirstName; file << student\_list[p].FirstName << ' ';  cout << "Отчество: "; cin >> student\_list[p].FatherName; file << student\_list[p].FatherName << ' ';  cout << "Дата поступления: "; cin >> student\_list[p].date; file << student\_list[p].date << ' ';  cout << "Специальность: "; cin >> student\_list[p].major; file << student\_list[p].major << ' ';  cout << "Группа: "; cin >> student\_list[p].group; file << student\_list[p].group << ' ';  cout << "Факультет: "; cin >> student\_list[p].Fac; file << student\_list[p].Fac << ' ';  cout << "Средний балл: "; cin >> student\_list[p].gpa; file << student\_list[p].gpa << ' '<<endl;  }  current\_size += number;  file.close();    }  void output() {  string buf;  ifstream file("database.txt");//открываем файл в режиме чтения  if (file.fail())  {  cout << "\n Ошибка открытия файла";  exit(1);  }  for (int i = 0; i < current\_size; i++) {//выводим содержимое файла  getline(file, buf);  cout << i + 1 << ". " << buf << endl;  }  file.close();    }  void find\_lastname(string fio) {  bool succes = false;  fill();//заполняем массив точными данными из файла  for (int i = 0; i < current\_size; i++) {//ищем совпадения фамилий и выводим все параметры структуры,если их находим  if (fio == student\_list[i].LastName) { succes = true; cout << endl << student\_list[i].LastName << ' ' << student\_list[i].FirstName << ' ' << student\_list[i].FatherName << ' ' << student\_list[i].date << ' ' << student\_list[i].major << ' ' << student\_list[i].group << ' ' << student\_list[i].Fac << ' ' << student\_list[i].gpa << endl; }  }  if (!succes) { cout << "Совпадений не найдено." << endl; }  }  int currentsize() {  int count = 0; string buf;  ifstream file("database.txt");  if (file.fail())  {  cout << "\n Ошибка открытия файла";  exit(1);  }  while (getline(file,buf)) {//считаем сколько строк мы смогли взять из файла,чтобы знать сколько студентов там есть  count++;  }  file.close();  return count;  }  void del(){  fill();//заполняем массив точными данными из файла  int number;  cout << "Введите номер студента,запись которого вы хотите удалить из реестра, на данный момент записано " << current\_size << " : "; cin >> number;  if (current\_size < number) { cout << "Данной записи не найдено."; exit(1); }  for (int i = number - 1; i < current\_size-1; i++) {//удаляем нужную строку из массива и сторчки выше спускаем на 1 позицую вниз  student\_list[i] = student\_list[i + 1];  }  student\_list[current\_size - 1] = {};  current\_size--;//корректируем размеры массива после всех манипуляций с ней  ofstream file("database.txt");//удаляем все прошлое содержимое,чтобы заменить его на актульные данные из массива  if (file.fail())  {  cout << "\n Ошибка открытия файла";  exit(1);  }  for (int p = 0; p < current\_size; p++)  {  file << student\_list[p].LastName << ' ';  file << student\_list[p].FirstName << ' ';  file << student\_list[p].FatherName << ' ';  file << student\_list[p].date << ' ';  file << student\_list[p].major << ' ';  file << student\_list[p].group << ' ';  file << student\_list[p].Fac << ' ';  file << student\_list[p].gpa << ' ' << endl;  }  file.close();  }  void fill() {  int number,pos=0; string buf;  ifstream file("database.txt");  if (file.fail())  {  cout << "\n Ошибка открытия файла";  exit(1);  }  for (int i = 0; i < current\_size; i++) {//данные о студенте представлют 1 строку с разделяющими пробелами,поэтому для получения данных в структуру мы должны извлекать все то,что находится между пробелами  getline(file, buf);  student\_list[i].LastName = buf.substr(pos, buf.find(' ')); pos = student\_list[i].LastName.size()+1;  student\_list[i].FirstName = buf.substr(pos, buf.find(' ',pos) - pos); pos += student\_list[i].FirstName.size() + 1;  student\_list[i].FatherName = buf.substr(pos, buf.find(' ', pos) - pos); pos += student\_list[i].FatherName.size() + 1;  student\_list[i].date = buf.substr(pos, buf.find(' ', pos) - pos); pos += student\_list[i].date.size() + 1;  student\_list[i].major = buf.substr(pos, buf.find(' ', pos) - pos); pos += student\_list[i].major.size() + 1;  student\_list[i].group = buf.substr(pos, buf.find(' ', pos) - pos); pos += student\_list[i].group.size() + 1;  student\_list[i].Fac = buf.substr(pos, buf.find(' ', pos) - pos); pos += student\_list[i].Fac.size() + 1;  student\_list[i].gpa = buf.substr(pos, buf.find(' ', pos) - pos);  pos = 0;  }  file.close();  } |
| **Результат программы** |
|  |

Дополнительные задания

1. Определить структуру для представления информации о сданных студентом экзаменах, содержащую поля: ФИО студента, число экзаменов, полученные оценки. Определить функции для обработки отдельного объекта (например, для проверки, сданы ли все экзамены на 4 и 5). Написать функцию для обработки массива структур. В результате обработки требуется вычислить характеристику успеваемости студентов, то есть отношение числа студентов, сдавших экзамены на 4 и 5, к общему числу студентов, в процентах.

|  |
| --- |
| **Код программы** |
| #include <iostream>  #include <string>  # define str\_len 30  # define size 30  using namespace std;  void enter\_new();  void good\_student\_check();  void out();  struct Student//создаем структуру студент,которая содержит нужные нам параметры и методы  {  string name;  int num\_ex;  int grades[10];  bool good\_student(int num\_ex,int grades[]) {//метод для определения является ли студент отлисником  for (int i=0; i < num\_ex; i++)  {  if (grades[i] < 4) { return false; }  }  return true;  }  };  struct Student list\_of\_student[size];//массив,позволяющий хранить данные о группе студентов в рамках одной переменной  int current\_size = 0; int choice;  int main()  {  system("color 70");  setlocale(LC\_CTYPE, "Russian");  cout << "Введите:" << endl;  cout << "1-для ввода данных" << endl;  cout << "2-для вывода записи(ей)" << endl;  cout << "3-проверка успеваимости студентов" << endl;  cout << "5-для выхода" << endl;  cin >> choice;    do//с помощью контекстного меню перемещаемся по возможностям программы  {  switch (choice)  {  case 1: enter\_new(); break;  case 2: out(); break;  case 3: good\_student\_check(); break;  default: cin >> choice;  }  } while (choice != 5);  }  void enter\_new()  {  cout << "Ввод информации" << endl;  if (current\_size < size)//отслеживаем заполненнось списка студентов  {  cout << "Строка номер ";  cout << current\_size + 1;  cout << endl << "ФИО: " << endl;  getline(cin, list\_of\_student[current\_size].name);  getline(cin, list\_of\_student[current\_size].name);  cout << "Кол-во экзаменов: " << endl;  cin >> list\_of\_student[current\_size].num\_ex;  cout << "Оценки: " << endl;  for (int i=0; i < list\_of\_student[current\_size].num\_ex; i++)  {  cin >> list\_of\_student[current\_size].grades[i];  }  current\_size++;  }  else  cout << "Введено максимальное кол-во строк";  cout << "Что дальше?" << endl;  cin >> choice;  }  void good\_student\_check()  {  int good\_student\_count = 0;  double gspa=0;  for (int i = 0; i < current\_size; i++)  {  if (list\_of\_student[i].good\_student(list\_of\_student[i].num\_ex, list\_of\_student[i].grades)) { good\_student\_count++; }//используя наш метод,мы ищем отличников среди группы  }  gspa += double(good\_student\_count) / double(current\_size);  cout << "Процент успевающих студентов в данной группе = " << gspa \* 100 << '%'<< endl;  cout << "Что дальше?" << endl;  cin >> choice;  }  void out()  {  int sw, n;  cout << "1-вывод 1 строки" << endl;  cout << "2-вывод всех строк" << endl;  cin >> sw;  if (sw == 1)  {  cout << "Номер выводимой строки " << endl; cin >> n; cout << endl;  cout << "ФИО: ";  cout << list\_of\_student[n - 1].name << endl;  cout << "Кол-во экзаменов: ";  cout << list\_of\_student[n - 1].num\_ex << endl;  cout << "Оценки: ";  for (int i=0; i < list\_of\_student[n - 1].num\_ex; i++)  {  cout << list\_of\_student[n - 1].grades[i] << ' ';  }  if (list\_of\_student[n - 1].good\_student(list\_of\_student[current\_size].num\_ex, list\_of\_student[current\_size].grades)) { cout <<endl<< "Отличник!"; }//с помощью метода структуры выводим является ли студент отличником  }  if (sw == 2)  {  for (int i = 0; i < current\_size; i++)  {  cout << "ФИО: ";  cout << list\_of\_student[i].name << endl;  cout << "Кол-во экзаменов: ";  cout << list\_of\_student[i].num\_ex << endl;  cout << "Оценки: ";  for (int j=0; j < list\_of\_student[i].num\_ex; j++)  {  cout << list\_of\_student[i].grades[j] << ' ';  }  if (list\_of\_student[i].good\_student(list\_of\_student[i].num\_ex, list\_of\_student[i].grades)) { cout << endl << "Отличник!"; }//с помощью метода структуры выводим является ли студент отличником  cout << endl << endl;  }  }  cout << "Что дальше?" << endl;  cin >> choice;  } |
| **Результат программы** |
|  |

2. Описать структуру с именем TRAIN, содержащую поля: названия пункта назначения, номер поезда, время отправления. Написать программу, выполняющую ввод с клавиатуры данных в массив, состоящий из восьми элементов типа TRAIN (записи должны быть размещены в алфавитном порядке по названиям пунктов назначения); вывод на экран информации о поездах, отправляющихся после введенного с клавиатуры времени (если таких поездов нет, то вывести сообщение об этом).

|  |
| --- |
| **Код программы** |
| #include <iostream>  #include <string>  # define size 8 //можно ввести до 8 составов  using namespace std;  void enter\_new();  void time\_check();  void out();  void rearrange();  struct train//создаем структуру поезд,которая содержит нужные нам параметры о поездах  {  string name;  int num;  int time\_h;  int time\_m;  };  train train\_list[size];//массив,позволяющий хранить данные о составах в рамках одной переменной  int current\_size = 0; int choice;  int main()  {  system("color 70");  setlocale(LC\_CTYPE, "Russian");  cout << "Введите:" << endl;  cout << "1-для ввода поездов" << endl;  cout << "2-для вывода записи(ей) о поездах" << endl;  cout << "3-проверка возможных для посадки поездов" << endl;  cout << "4-для выхода" << endl;  cin >> choice;  while (true)//с помощью контекстного меню перемещаемся по возможностям программы  {  switch (choice)  {  case 1: enter\_new(); break;  case 2: out(); break;  case 3: time\_check(); break;  case 4: return 0; break;  default: cout << "try again" << endl; cin >> choice;  }  }  }  void enter\_new()  {  if (current\_size < size)//отслеживаем заполненнось расписания поездов  {  cout << endl << "Место назначения: " << endl;  getline(cin, train\_list[current\_size].name);  getline(cin, train\_list[current\_size].name);  cout << "Номер состава: " << endl;  cin >> train\_list[current\_size].num;  cout << "Время отправления(ЧЧ ММ): " << endl;  cin >> train\_list[current\_size].time\_h>> train\_list[current\_size].time\_m;    current\_size++;  }  else  cout << "Введено максимальное кол-во поездов";  cout << "Что дальше?" << endl;  cin >> choice;  }  void time\_check() {  int hours, minutes;  cout << "Введите время вашего прибытия на вокзал: ";  cin >> hours >> minutes;  for (int i = 0; i < current\_size; i++)  {  if (train\_list[i].time\_h>=hours && train\_list[i].time\_m>=minutes) {//если поезд отправляется позже заданного времени,то мы его выводим  cout << i + 1 << ". " << train\_list[i].name << " \t" << train\_list[i].num << " \t" << train\_list[i].time\_h << ':' << train\_list[i].time\_m << " \t" << endl;  }  }  cout << "Что дальше?" << endl;  cin >> choice;  }  void out()  {  rearrange();//прежде чем выводить,используем фукцию для сортировки  cout << "Место назначения \t" << "Номер состава \t" << "Время отпрвления \t" << endl;    for (int i = 0; i < current\_size; i++)  {  cout << i + 1 << ". " << train\_list[i].name << " \t" << train\_list[i].num << " \t" << train\_list[i].time\_h << ':' << train\_list[i].time\_m << " \t" << endl;  }    cout << "Что дальше?" << endl;  cin >> choice;  }  void rearrange() {// данная функция выполняет сортировку данных в расписании по месту назначения  bool skip\_check = 0;  for (int i = 0; i < current\_size; i++) {  for (int j = 0; j < current\_size; j++) {  if (train\_list[i].name[0] < train\_list[j].name[0]) { swap(train\_list[i], train\_list[j]); skip\_check = 1; }//сравниваем первую букву в месте назначения и в зависимости от нее меняем положение поезда в расписании  }  if (skip\_check) { i--; skip\_check = 0; }  }  } |
| **Результат программы** |
|  |

3. Определить структуру для представления информации о наличии санаторных путевок, содержащую следующие поля: Название санатория, Место расположения, Лечебный профиль, Количество путевок. Представить введенные данные в виде таблицы, сгруппировав их по лечебным профилям санаториев. В пределах каждой группы данные отсортировать по названиям санаториев. Организовать поиск информации и вывод результатов.

|  |  |
| --- | --- |
| **Код программы** |  |
| #include <iostream>  #include <string>  # define size 15  using namespace std;  void enter\_new();//функция для ввода  void find\_resort(string);//функция для поиска информации  void out();//функция для вывода  void rearrange();//функция для сортировки по имени  bool profile\_check(int, string);//функция на определения появления нового лечебного профиля в списке  struct resort//создаем структуру поезд,которая содержит нужные нам параметры о поездах  {  string name;  string place;  string profile;  int vacant\_num;  };  resort resort\_list[size];//массив,позволяющий хранить данные о составах в рамках одной переменной  string resort\_profile[15];  int current\_size = 0, choice,counter\_prof=0;  int main()  {  string name;  system("color 70");  setlocale(LC\_CTYPE, "Russian");  cout << "Введите:" << endl;  cout << "1-для ввода санаториев" << endl;  cout << "2-для вывода записи(ей) о санаториях" << endl;  cout << "3-поиск информации по названию" << endl;  cout << "4-для выхода" << endl;  cin >> choice;  while (true)//с помощью контекстного меню перемещаемся по возможностям программы  {  switch (choice)  {  case 1: enter\_new(); break;  case 2: out(); break;  case 3:  cout << "Введите имя санатория:";  getline(cin, name); getline(cin, name);  find\_resort(name); break;  case 4: return 0; break;  default: cout << "try again" << endl; cin >> choice;  }  }  }  void enter\_new()  {  if (current\_size < size)//отслеживаем заполненнось расписания поездов  {  cout << endl << "Название: " << endl;  getline(cin, resort\_list[current\_size].name);  getline(cin, resort\_list[current\_size].name);  cout << endl << "Место расположения: " << endl;  getline(cin, resort\_list[current\_size].place);  cout << endl << "Лечебный профиль: " << endl;  getline(cin, resort\_list[current\_size].profile);  if(profile\_check((counter\_prof),resort\_list[current\_size].profile)){counter\_prof++;}//проверка вводился ли новый профиль ранее  cout << "Кол-во доступных путевок: " << endl;  cin >> resort\_list[current\_size].vacant\_num;    current\_size++;  }  else  cout << "Введено максимальное кол-во поездов";  cout << "Что дальше?" << endl;  cin >> choice;  }    void out()  {  rearrange();//прежде чем выводить,используем фукцию для сортировки  cout << "Название \t" << "Место расположения \t" << "Лечебный профиль \t" << "Кол-во доступных путевок" << endl;    int count\_line = 1;  for (int i = 0; i < counter\_prof; i++)  {  cout << endl << resort\_profile[i] << endl;//выводим санатории по профилям  for (int j = 0; j < current\_size; j++)  {  if (resort\_profile[i]==resort\_list[j].profile)  {  cout << count\_line << ". " << resort\_list[j].name << " \t" << resort\_list[j].place << " \t" << resort\_list[j].profile << ':' << resort\_list[j].vacant\_num << " \t" << endl;  }  }  count\_line = 1;  }    cout << "Что дальше?" << endl;  cin >> choice;  }  void rearrange() {// данная функция выполняет сортировку данных в расписании по месту назначения  bool skip\_check = 0;  for (int i = 0; i < current\_size; i++) {  for (int j = 0; j < current\_size; j++) {  if (resort\_list[i].name[0] < resort\_list[j].name[0]) { swap(resort\_list[i], resort\_list[j]); skip\_check = 1; }//сравниваем первую букву в месте назначения и в зависимости от нее меняем положение поезда в расписании  }  if (skip\_check) { i--; skip\_check = 0; }  }  }  bool profile\_check(int count,string prof) {  bool identical = 0;  for (int i = 0; i < count; i++)  {  if (resort\_profile[i] == prof)  {  identical = 1;  }  }  if (!identical) {//если ранее профиль не появляся в массиве профилей,то добовляем его в первую свободную позицию  resort\_profile[count] = prof;  return true;  }  else { return false; }  }  void find\_resort(string name) {  bool succes = false;  for (int i = 0; i < current\_size; i++) {//ищем совпадения фамилий и выводим все параметры структуры,если их находим  if (name == resort\_list[i].name) { succes = true; cout << resort\_list[i].name << " \t" << resort\_list[i].place << " \t" << resort\_list[i].profile << ':' << resort\_list[i].vacant\_num << " \t" << endl; }  }  if (!succes) { cout << "Совпадений не найдено." << endl; }  cout << "Что дальше?" << endl;  cin >> choice;  } |  |
| **Результат программы** |  |
|  |  |