**Лабораторная работа № 5. Объединения, перечисления, битовые поля**

|  |  |  |
| --- | --- | --- |
| 1. Изучить принципы работы с *объединениями*, выполнив программу, записанную в данном пункте. | #include <iostream>  using namespace std;  union Utypes  {  char c;  int ivalue;  float fvalue;  double dvalue;  } mun;  int main()  {  setlocale(LC\_ALL, "Russian");  mun.c = 'b';  cout << mun.c << endl;  mun.ivalue = 1990;  cout << mun.ivalue << endl;  mun.fvalue = 19.90;  cout << mun.fvalue << endl;  mun.dvalue = 987654.32E+13;  cout << mun.dvalue << endl;  cout << mun.c << endl;  cout << mun.ivalue << endl;  cout << mun.fvalue << endl;  cout << mun.dvalue << endl;  cout << "Размер объединения составляет " << sizeof(Utypes) << " байт.";  return 0;  } |  |
| 2. Изучить принципы работы с *перечислениями*, выполнив программу, записанную в данном пункте. | #include <iostream>  using namespace std;  enum Emonths  {  January = 1, February, March,  April, May, June, July, August, September,  October, November, December  } months;  int main(void)  {  setlocale(LC\_CTYPE, "Russian");  int current\_month, left;  cout << "Введите номер текущего месяца (от 1 до 12):";  cin >> current\_month;  months = December;  left = (int)months - current\_month;  cout << "\nДо конца года осталось " << left << " месяца(ев)\n";  return 0;  } |  |
| 3. Изучить принципы работы с *битовыми полями*, выполнив программу, записанную в данном пункте. | #include <iostream>  using namespace std;  struct byte  {  unsigned a : 1;  unsigned b : 1;  unsigned c : 1;  unsigned d : 1;  unsigned e : 1;  unsigned f : 1;  unsigned g : 1;  unsigned h : 1;  };  union bits  {  char ch;  struct byte bit;  } ascii;  void disp\_bits(bits b);  void main()  {  do  {  cin >> ascii.ch;  disp\_bits(ascii);  } while (ascii.ch != 'q'); //выход при вводе q  }  void disp\_bits(bits b)  {  if (b.bit.h) cout << "1"; else cout << "0";  if (b.bit.g) cout << "1"; else cout << "0";  if (b.bit.f) cout << "1"; else cout << "0 ";  if (b.bit.e) cout << "1"; else cout << "0";  if (b.bit.d) cout << "1"; else cout << "0";  if (b.bit.c) cout << "1"; else cout << "0";  if (b.bit.b) cout << "1"; else cout << "0";  if (b.bit.a) cout << "1"; else cout << "0";  cout << "\n";  } |  |
| **Вариант 10 Государство**. Наименование, столица, численность населения, площадь. Выбор государства по столице. Форму правления реализовать с помощью перечисления. | | |
| #include <iostream>  #include <iomanip>  using namespace std;  enum polit { monarchy, republic, dict, oligarchy, totalit };  struct GOSUDAR {  char name[30];  int population;  int area;  char capital[30];  polit form;  };  void handleEnum(char temp[50], GOSUDAR all[], int size) {  if (strcmp("monarchy", temp) == 0) { all[size].form = monarchy; }  if (strcmp("republic", temp) == 0) { all[size].form = republic; }  if (strcmp("dict", temp) == 0) { all[size].form = dict; }  if (strcmp("oligarchy", temp) == 0) { all[size].form = oligarchy; }  if (strcmp("totalit", temp) == 0) { all[size].form = totalit; }  }  const char\* writeEnum(enum polit en, char temp[50]) {  switch (en) {  case monarchy: {  strcpy\_s(temp, 50, "monarchy");  break;  }  case republic: {  strcpy\_s(temp, 50, "republic");  break;  }  case dict: {  strcpy\_s(temp, 50, "dict");  break;  }  case oligarchy: {  strcpy\_s(temp, 50, "oligarchy");  break;  }  case totalit: {  strcpy\_s(temp, 50, "totalit");  break;  }  default: {  strcpy\_s(temp, 50, "unknown");  break;  }  }  return temp;  }  void readMas(GOSUDAR all[20], int& size) {  char temp[20];  cout << left << setw(15) << "Имя" << setw(10) << "Население" << setw(15) << "Площадь" << setw(15) << "Столица" << setw(20) << "Форма правления" << endl;  cin >> all[size].name >> all[size].population >> all[size].area >> all[size].capital >> temp;  handleEnum(temp, all, size);  size++;  }  void writeMas(GOSUDAR all[20], int size) {  cout << left << setw(15) << "Имя" << setw(10) << "Население" << setw(15) << "Площадь" << setw(15) << "Столица" << setw(20) << "Форма правления" << endl;  for (int i = 0; i < size; i++) {  char temp[50];  writeEnum(all[i].form, temp);  cout << left << setw(15) << all[i].name << setw(10) << all[i].population << setw(15) << all[i].area << setw(15) << all[i].capital << setw(20) << temp << endl;  }  }  int searchMas(GOSUDAR all[20], int size, const char\* item) {  for (int i = 0; i < size; i++) {  bool flag = true;  for (int j = 0; item[j] != '\0' && all[i].capital[j] != '\0'; j++) {  if (item[j] != all[i].capital[j]) {  flag = false;  }  }  if (flag) {  return i;  }  }  return -1;  }  void deleteElement(GOSUDAR all[20], int& size, int nomer) {  for (int i = nomer; i < size - 1; i++) {  all[i] = all[i + 1];  }  size--;  }  int main() {  setlocale(LC\_ALL, "rus");  int choice;  GOSUDAR all[20];  int size = 0;  do {  cout << "\n1.Ввод данных с клавиатуры и запись в массив\n";  cout << "2.Вывод данных из массива\n";  cout << "3.Поиск по столице\n";  cout << "4.Удаление данных из массива\n";  cout << "0.Выход из программы\n\n";  cout << "Введите номер операции: ";  cin >> choice;  cin.ignore();  int ans = 0;  switch (choice) {  case 1: {  readMas(all, size);  break;  }  case 2: {  writeMas(all, size);  break;  }  case 3: {  cout << "Введите столицу\n";  char buf[256];  cin.getline(buf, 256);  int ans = searchMas(all, size, buf);  if (ans == -1) {  cout << "Элемент не найден\n";  }  else {  cout << "Элемент номер: " << ans + 1 << '\n';  }  break;  }  case 4: {  cout << "Введите номер элемента для удаления: ";  int k;  cin >> k;  deleteElement(all, size, k - 1);  cout << "Элемент удалён\n";  break;  }  case 0:  break;  default: {  cout << "Неверный ввод\n";  break;  }  }  } while (choice != 0);  return 0;  } | |  |
| **Задание 2:Горожанин.** Ф.И.О., дата рождения, адрес, пол (м, ж). Реализовать выборку по году рождения. | | |
| #include <iostream>  #include <fstream>  #include <iomanip>  #include <cstring>  using namespace std;  union Citizen {  char lastName[50];  char firstName[50];  char middleName[50];  char birthDate[15];  char address[100];  char gender;  };  void writeToFile(const char\* filename) {  ofstream fout(filename);  if (!fout) {  cerr << "Ошибка открытия файла!" << endl;  return;  }  Citizen citizen;  char lastName[50], firstName[50], middleName[50], birthDate[15], address[100];  char gender;  cout << "Введите фамилию: ";  cin.ignore();  cin >> lastName;  strncpy\_s(citizen.lastName, lastName, sizeof(citizen.lastName) - 1);  fout << citizen.lastName << "\t";  cout << "Введите имя: ";  cin >> firstName;  strncpy\_s(citizen.firstName, firstName, sizeof(citizen.firstName) - 1);  fout << citizen.firstName << "\t";  cout << "Введите отчество: ";  cin >> middleName;  strncpy\_s(citizen.middleName, middleName, sizeof(citizen.middleName) - 1);  fout << citizen.middleName << "\t";  cout << "Введите дату рождения (ДД.ММ.ГГГГ): ";  cin >> birthDate;  strncpy\_s(citizen.birthDate, birthDate, sizeof(citizen.birthDate) - 1);  fout << citizen.birthDate << "\t";  cout << "Введите адрес: ";  cin >> address;  strncpy\_s(citizen.address, address, sizeof(citizen.address) - 1);  fout << citizen.address << "\t";  cout << "Введите пол (м/ж): ";  cin >> gender;  citizen.gender = gender;  fout << citizen.gender << '\n';  fout.close();  }  void readFromFile(const char\* filename) {  ifstream fin(filename);  if (!fin) {  cerr << "Ошибка открытия файла!" << endl;  return;  }  Citizen citizen;  cout << left << setw(20) << "Фамилия" << setw(20) << "Имя" << setw(20) << "Отчество"  << setw(15) << "Дата рождения" << setw(40) << "Адрес" << setw(5) << "Пол" << endl;  while (!fin.fail()) {  fin >> citizen.lastName;  if (fin.fail()) break;  char lastName[50];  strncpy\_s(lastName, citizen.lastName, sizeof(lastName) - 1);  fin >> citizen.firstName;  if (fin.fail()) break;  char firstName[50];  strncpy\_s(firstName, citizen.firstName, sizeof(firstName) - 1);  fin >> citizen.middleName;  if (fin.fail()) break;  char middleName[50];  strncpy\_s(middleName, citizen.middleName, sizeof(middleName) - 1);  fin >> citizen.birthDate;  if (fin.fail()) break;  char birthDate[15];  strncpy\_s(birthDate, citizen.birthDate, sizeof(birthDate) - 1);  fin >> citizen.address;  if (fin.fail()) break;  char address[100];  strncpy\_s(address, citizen.address, sizeof(address) - 1);  fin >> citizen.gender;  if (fin.fail()) break;  char gender = citizen.gender;    cout << left << setw(20) << lastName << setw(20) << firstName << setw(20) << middleName  << setw(15) << birthDate << setw(40) << address << setw(5) << gender << endl;  }  fin.close();  }  void searchByYear(const char\* filename, const char\* year) {  ifstream fin(filename);  if (!fin) {  cerr << "Ошибка открытия файла!" << endl;  return;  }  Citizen citizen;  bool found = false;  cout << "Результаты поиска по году " << year << ":\n";    while (!fin.fail()) {  fin >> citizen.lastName;  if (fin.fail()) break;  char lastName[50];  strncpy\_s(lastName, citizen.lastName, sizeof(lastName) - 1);  fin >> citizen.firstName;  if (fin.fail()) break;  char firstName[50];  strncpy\_s(firstName, citizen.firstName, sizeof(firstName) - 1);  fin >> citizen.middleName;  if (fin.fail()) break;  char middleName[50];  strncpy\_s(middleName, citizen.middleName, sizeof(middleName) - 1);  fin >> citizen.birthDate;  if (fin.fail()) break;  char birthDate[15];  strncpy\_s(birthDate, citizen.birthDate, sizeof(birthDate) - 1);  fin >> citizen.address;  if (fin.fail()) break;  char address[100];  strncpy\_s(address, citizen.address, sizeof(address) - 1);  fin >> citizen.gender;  if (fin.fail()) break;  char gender = citizen.gender;    if (strstr(birthDate, year)) {  cout << left << setw(20) << lastName << setw(20) << firstName << setw(20) << middleName  << setw(15) << birthDate << setw(40) << address << setw(5) << gender << endl;  found = true;  }  }  if (!found) {  cout << "Совпадений не найдено." << endl;  }  fin.close();  }  int main() {  setlocale(LC\_ALL, "rus");  const char filename[] = "citizens.txt";  int choice;  do {  cout << "\n1. Ввод данных с клавиатуры и запись в файл" << endl;  cout << "2. Вывод данных из файла" << endl;  cout << "3. Поиск по году рождения" << endl;  cout << "0. Выход" << endl;  cout << "Введите номер операции: ";  cin >> choice;  cin.ignore();  switch (choice) {  case 1:  writeToFile(filename);  break;  case 2:  readFromFile(filename);  break;  case 3: {  char year[5];  cout << "Введите год для поиска: ";  cin >> year;  searchByYear(filename, year);  break;  }  case 0:  break;  default:  cout << "Неверный ввод!" << endl;  }  } while (choice != 0);  return 0;  } | |  |

Дополнительные задания

|  |  |
| --- | --- |
| **Вариант 12:**  **Записная книжка.** Ф.И.О, дата рождения, адрес, телефон . Поиск по фамилии. Дату рождения реализовать с помощью битового поля. | |
| #include <iostream>  #include <iomanip>  #include <cstring>  using namespace std;  struct Date {  unsigned int day : 5;  unsigned int month : 4;  unsigned int year : 12;  };  struct Contact {  char fullName[50];  Date birthDate;  char address[100];  char phone[20];  };  void readContact(Contact contacts[], int& size) {  cout << "Введите Ф.И.О.: ";  cin.getline(contacts[size].fullName, 50);  cout << "Введите дату рождения (день месяц год): ";  unsigned int d, m, y;  cin >> d >> m >> y;  contacts[size].birthDate = { d, m, y };  cin.ignore();  cout << "Введите адрес: ";  cin.getline(contacts[size].address, 100);  cout << "Введите телефон: ";  cin.getline(contacts[size].phone, 20);  size++;  }  void writeContacts(Contact contacts[], int size) {  cout << left << setw(30) << "Ф.И.О." << setw(15) << "Дата рождения" << setw(30) << "Адрес" << setw(15) << "Телефон" << endl;  for (int i = 0; i < size; i++) {  cout << left << setw(30) << contacts[i].fullName << setw(2) << contacts[i].birthDate.day << "." << setw(2) << contacts[i].birthDate.month << "." << setw(10) << contacts[i].birthDate.year << setw(30) << contacts[i].address << setw(15) << contacts[i].phone << endl;  }  }  int searchContact(Contact contacts[], int size, const char\* lastName) {  for (int i = 0; i < size; i++) {  if (strncmp(contacts[i].fullName, lastName, strlen(lastName)) == 0) {  return i;  }  }  return -1;  }  void deleteContact(Contact contacts[], int& size, int index) {  for (int i = index; i < size - 1; i++) {  contacts[i] = contacts[i + 1];  }  size--;  }  int main() {  setlocale(LC\_ALL, "rus");  Contact contacts[20];  int size = 0, choice;  do {  cout << "\n1. Добавить контакт\n";  cout << "2. Вывести контакты\n";  cout << "3. Поиск по фамилии\n";  cout << "4. Удалить контакт\n";  cout << "0. Выход\n";  cout << "Выберите операцию: ";  cin >> choice;  cin.ignore();  switch (choice) {  case 1:  readContact(contacts, size);  break;  case 2:  writeContacts(contacts, size);  break;  case 3: {  cout << "Введите фамилию для поиска: ";  char lastName[50];  cin.getline(lastName, 50);  int index = searchContact(contacts, size, lastName);  if (index == -1) {  cout << "Контакт не найден\n";  }  else {  cout << "Контакт найден: " << contacts[index].fullName << "\n";  }  break;  }  case 4:  cout << "Введите номер контакта для удаления: ";  int num;  cin >> num;  if (num > 0 && num <= size) {  deleteContact(contacts, size, num - 1);  cout << "Контакт удален\n";  }  else {  cout << "Некорректный номер\n";  }  break;  case 0:  break;  default:  cout << "Неверный ввод\n";  break;  }  } while (choice != 0);  return 0;  } |  |
| **Записная книжка.** Ф.И.О, дата рождения, адрес, телефон, место работы или учебы, должность. Поиск по фамилии. | |
| #include <iostream>  #include <fstream>  #include <iomanip>  #include <cstring>  using namespace std;  union NotebookEntry {  char lastName[50];  char firstName[50];  char middleName[50];  char birthDate[15];  char address[100];  char phone[20];  char workplace[50];  char position[50];  };  void writeToFile(const char\* filename) {  ofstream fout(filename);  if (!fout) {  cerr << "Ошибка открытия файла!" << endl;  return;  }  NotebookEntry entry;  char lastName[50], firstName[50], middleName[50], birthDate[15], address[100], phone[20], workplace[50], position[50];  cout << "Введите фамилию: ";  cin >> lastName;  strncpy\_s(entry.lastName, lastName, sizeof(entry.lastName) - 1);  fout << entry.lastName << "\t";  cout << "Введите имя: ";  cin >> firstName;  strncpy\_s(entry.firstName, firstName, sizeof(entry.firstName) - 1);  fout << entry.firstName << "\t";  cout << "Введите отчество: ";  cin >> middleName;  strncpy\_s(entry.middleName, middleName, sizeof(entry.middleName) - 1);  fout << entry.middleName << "\t";  cout << "Введите дату рождения (ДД.ММ.ГГГГ): ";  cin >> birthDate;  strncpy\_s(entry.birthDate, birthDate, sizeof(entry.birthDate) - 1);  fout << entry.birthDate << "\t";  cout << "Введите адрес: ";  cin >> address;  strncpy\_s(entry.address, address, sizeof(entry.address) - 1);  fout << entry.address << "\t";  cout << "Введите телефон: ";  cin >> phone;  strncpy\_s(entry.phone, phone, sizeof(entry.phone) - 1);  fout << entry.phone << "\t";  cout << "Введите место работы/учебы: ";  cin >> workplace;  strncpy\_s(entry.workplace, workplace, sizeof(entry.workplace) - 1);  fout << entry.workplace << "\t";  cout << "Введите должность: ";  cin >> position;  strncpy\_s(entry.position, position, sizeof(entry.position) - 1);  fout << entry.position << '\n';  fout.close();  }  void readFromFile(const char\* filename) {  ifstream fin(filename);  if (!fin) {  cerr << "Ошибка открытия файла!" << endl;  return;  }  NotebookEntry entry;  cout << left << setw(20) << "Фамилия" << setw(20) << "Имя" << setw(20) << "Отчество"  << setw(15) << "Дата рождения" << setw(25) << "Адрес" << setw(15) << "Телефон"  << setw(25) << "Место работы/учебы" << setw(20) << "Должность" << endl;  while (!fin.fail()) {  fin >> entry.lastName;  if (fin.fail()) break;  char lastName[50];  strncpy\_s(lastName, entry.lastName, sizeof(lastName) - 1);  fin >> entry.firstName;  if (fin.fail()) break;  char firstName[50];  strncpy\_s(firstName, entry.firstName, sizeof(firstName) - 1);  fin >> entry.middleName;  if (fin.fail()) break;  char middleName[50];  strncpy\_s(middleName, entry.middleName, sizeof(middleName) - 1);  fin >> entry.birthDate;  if (fin.fail()) break;  char birthDate[15];  strncpy\_s(birthDate, entry.birthDate, sizeof(birthDate) - 1);  fin >> entry.address;  if (fin.fail()) break;  char address[100];  strncpy\_s(address, entry.address, sizeof(address) - 1);  fin >> entry.phone;  if (fin.fail()) break;  char phone[20];  strncpy\_s(phone, entry.phone, sizeof(phone) - 1);  fin >> entry.workplace;  if (fin.fail()) break;  char workplace[50];  strncpy\_s(workplace, entry.workplace, sizeof(workplace) - 1);  fin >> entry.position;  if (fin.fail()) break;  char position[50];  strncpy\_s(position, entry.position, sizeof(position) - 1);  cout << left << setw(20) << lastName << setw(20) << firstName << setw(20) << middleName  << setw(15) << birthDate << setw(25) << address << setw(15) << phone  << setw(25) << workplace << setw(20) << position << endl;  }  fin.close();  }  void searchByLastName(const char\* filename, const char\* searchLastName) {  ifstream fin(filename);  if (!fin) {  cerr << "Ошибка открытия файла!" << endl;  return;  }  NotebookEntry entry;  bool found = false;  int i = 0;  while (!fin.fail()) {  i++;  fin >> entry.lastName;  if (fin.fail()) break;  if (strcmp(entry.lastName, searchLastName) == 0) {  fin >> entry.firstName >> entry.middleName >> entry.birthDate >> entry.address >> entry.phone >> entry.workplace >> entry.position;  cout << "Запись найдена: " << i;  found = true;  }  }  if (!found) {  cout << "Совпадений не найдено." << endl;  }  fin.close();  }  int main() {  setlocale(LC\_ALL, "rus");  const char filename[] = "notebook.txt";  int choice;  do {  cout << "\n1. Ввод данных с клавиатуры и запись в файл" << endl;  cout << "2. Вывод данных из файла" << endl;  cout << "3. Поиск по фамилии" << endl;  cout << "0. Выход" << endl;  cout << "Введите номер операции: ";  cin >> choice;  switch (choice) {  case 1:  writeToFile(filename);  break;  case 2:  readFromFile(filename);  break;  case 3: {  char searchLastName[50];  cout << "Введите фамилию для поиска: ";  cin >> searchLastName;  searchByLastName(filename, searchLastName);  break;  }  case 0:  break;  default:  cout << "Неверный ввод!" << endl;  }  } while (choice != 0);  return 0;  } |  |

**Вариант 14**

|  |  |
| --- | --- |
| **Список клиентов гостиницы.** Паспортные данные, даты приезда и отъезда, номер, тип размещения (люкс, одноместный, двухместный, трехместный, апартаменты). Поиск гостя по дате приезда. Даты приезда и отъезда реализовать с помощью битового поля, тип размещения − с помощью перечисления. | |
| #include <iostream>  #include <iomanip>  #include <cstring>  using namespace std;  struct Date {  unsigned day : 5;  unsigned month : 4;  unsigned year : 12;  };  enum RoomType { LUXURY, SINGLE, DOUBLE, TRIPLE, APARTMENT };  struct Guest {  char passportData[50];  Date arrivalDate;  Date departureDate;  int roomNumber;  RoomType roomType;  };  void handleEnum(char temp[50], Guest guests[], int size) {  if (strcmp("luxury", temp) == 0) { guests[size].roomType = LUXURY; }  if (strcmp("single", temp) == 0) { guests[size].roomType = SINGLE; }  if (strcmp("double", temp) == 0) { guests[size].roomType = DOUBLE; }  if (strcmp("triple", temp) == 0) { guests[size].roomType = TRIPLE; }  if (strcmp("apartment", temp) == 0) { guests[size].roomType = APARTMENT; }  }  const char\* writeEnum(RoomType type, char temp[50]) {  switch (type) {  case LUXURY: strcpy\_s(temp, 50, "luxury"); break;  case SINGLE: strcpy\_s(temp, 50, "single"); break;  case DOUBLE: strcpy\_s(temp, 50, "double"); break;  case TRIPLE: strcpy\_s(temp, 50, "triple"); break;  case APARTMENT: strcpy\_s(temp, 50, "apartment"); break;  default: strcpy\_s(temp, 50, "unknown"); break;  }  return temp;  }  void readGuest(Guest guests[], int& size) {  cout << "Введите паспортные данные: ";  cin.getline(guests[size].passportData, 50);  cout << "Введите дату приезда (день месяц год): ";  unsigned int d, m, y;  cin >> d >> m >> y;  guests[size].arrivalDate = { d,m,y };  cout << "Введите дату отъезда (день месяц год): ";  cin >> d >> m >> y;  guests[size].departureDate= { d,m,y };  cout << "Введите номер комнаты: ";  cin >> guests[size].roomNumber;  cin.ignore();  cout << "Введите тип размещения (luxury, single, double, triple, apartment): ";  char temp[50];  cin.getline(temp, 50);  handleEnum(temp, guests, size);  size++;  }  void writeGuests(Guest guests[], int size) {  cout << left << setw(20) << "Паспортные данные" << setw(15) << "Дата приезда" << setw(15) << "Дата отъезда" << setw(10) << "Комната" << setw(15) << "Тип" << endl;  for (int i = 0; i < size; i++) {  char temp[50];  cout << left << setw(20) << guests[i].passportData  << setw(2) << guests[i].arrivalDate.day << '.' << setw(2) << guests[i].arrivalDate.month << '.' << setw(8) << guests[i].arrivalDate.year  << setw(2) << guests[i].departureDate.day << '.' << setw(2) << guests[i].departureDate.month << '.' << setw(8) << guests[i].departureDate.year  << setw(10) << guests[i].roomNumber  << setw(15) << writeEnum(guests[i].roomType, temp) << endl;  }  }  int searchGuest(Guest guests[], int size, Date date) {  for (int i = 0; i < size; i++) {  if (guests[i].arrivalDate.day == date.day && guests[i].arrivalDate.month == date.month && guests[i].arrivalDate.year == date.year) {  return i;  }  }  return -1;  }  int main() {  setlocale(LC\_ALL, "rus");  Guest guests[20];  int size = 0, choice;  do {  cout << "\n1. Добавить гостя\n";  cout << "2. Вывести гостей\n";  cout << "3. Поиск гостя по дате приезда\n";  cout << "0. Выход\n";  cout << "Выберите операцию: ";  cin >> choice;  cin.ignore();  switch (choice) {  case 1:  readGuest(guests, size);  break;  case 2:  writeGuests(guests, size);  break;  case 3: {  cout << "Введите дату приезда (день месяц год): ";    unsigned int d, m, y;  cin >> d >> m >> y;  Date date = { d,m,y };  int index = searchGuest(guests, size, date);  if (index == -1) {  cout << "Гость не найден\n";  }  else {  cout << "Гость номер: " << index+1 << "\n";  }  break;  }  case 0:  break;  default:  cout << "Неверный ввод\n";  break;  }  } while (choice != 0);  return 0;  } |  |
| **Список клиентов гостиницы.** Паспортные данные, даты приезда и отъезда, номер, тип размещения (люкс, одноместный, двухместный, трехместный, апартаменты). Поиск гостя по фамилии. | |
| #include <iostream>  #include <fstream>  #include <iomanip>  #include <cstring>  using namespace std;  union HotelGuest {  char passportData[50];  char lastName[50];  char firstName[50];  char middleName[50];  char checkInDate[15];  char checkOutDate[15];  char roomNumber[10];  char roomType[20];  };  void writeToFile(const char\* filename) {  ofstream fout(filename);  if (!fout) {  cerr << "Ошибка открытия файла!" << endl;  return;  }  HotelGuest guest;  char passportData[50], lastName[50], firstName[50], middleName[50], checkInDate[15], checkOutDate[15], roomNumber[10], roomType[20];  cout << "Введите паспортные данные: ";  cin >> passportData;  strncpy\_s(guest.passportData, passportData, sizeof(guest.passportData) - 1);  fout << guest.passportData << "\t";  cout << "Введите фамилию: ";  cin >> lastName;  strncpy\_s(guest.lastName, lastName, sizeof(guest.lastName) - 1);  fout << guest.lastName << "\t";  cout << "Введите имя: ";  cin >> firstName;  strncpy\_s(guest.firstName, firstName, sizeof(guest.firstName) - 1);  fout << guest.firstName << "\t";  cout << "Введите отчество: ";  cin >> middleName;  strncpy\_s(guest.middleName, middleName, sizeof(guest.middleName) - 1);  fout << guest.middleName << "\t";  cout << "Введите дату приезда (ДД.ММ.ГГГГ): ";  cin >> checkInDate;  strncpy\_s(guest.checkInDate, checkInDate, sizeof(guest.checkInDate) - 1);  fout << guest.checkInDate << "\t";  cout << "Введите дату отъезда (ДД.ММ.ГГГГ): ";  cin >> checkOutDate;  strncpy\_s(guest.checkOutDate, checkOutDate, sizeof(guest.checkOutDate) - 1);  fout << guest.checkOutDate << "\t";  cout << "Введите номер комнаты: ";  cin >> roomNumber;  strncpy\_s(guest.roomNumber, roomNumber, sizeof(guest.roomNumber) - 1);  fout << guest.roomNumber << "\t";  cout << "Введите тип размещения (люкс, одноместный, двухместный, трехместный, апартаменты): ";  cin >> roomType;  strncpy\_s(guest.roomType, roomType, sizeof(guest.roomType) - 1);  fout << guest.roomType << '\n';  fout.close();  }  void readFromFile(const char\* filename) {  ifstream fin(filename);  if (!fin) {  cerr << "Ошибка открытия файла!" << endl;  return;  }  HotelGuest guest;  cout << left << setw(20) << "Паспортные данные" << setw(20) << "Фамилия" << setw(20) << "Имя"  << setw(20) << "Отчество" << setw(15) << "Дата приезда" << setw(15) << "Дата отъезда"  << setw(10) << "Номер" << setw(20) << "Тип размещения" << endl;  while (!fin.fail()) {  fin >> guest.passportData;  if (fin.fail()) break;  char passportData[50];  strncpy\_s(passportData, guest.passportData, sizeof(passportData) - 1);  fin >> guest.lastName;  if (fin.fail()) break;  char lastName[50];  strncpy\_s(lastName, guest.lastName, sizeof(lastName) - 1);  fin >> guest.firstName;  if (fin.fail()) break;  char firstName[50];  strncpy\_s(firstName, guest.firstName, sizeof(firstName) - 1);  fin >> guest.middleName;  if (fin.fail()) break;  char middleName[50];  strncpy\_s(middleName, guest.middleName, sizeof(middleName) - 1);  fin >> guest.checkInDate;  if (fin.fail()) break;  char checkInDate[15];  strncpy\_s(checkInDate, guest.checkInDate, sizeof(checkInDate) - 1);  fin >> guest.checkOutDate;  if (fin.fail()) break;  char checkOutDate[15];  strncpy\_s(checkOutDate, guest.checkOutDate, sizeof(checkOutDate) - 1);  fin >> guest.roomNumber;  if (fin.fail()) break;  char roomNumber[10];  strncpy\_s(roomNumber, guest.roomNumber, sizeof(roomNumber) - 1);  fin >> guest.roomType;  if (fin.fail()) break;  char roomType[20];  strncpy\_s(roomType, guest.roomType, sizeof(roomType) - 1);  cout << left << setw(20) << passportData << setw(20) << lastName << setw(20) << firstName  << setw(20) << middleName << setw(15) << checkInDate << setw(15) << checkOutDate  << setw(10) << roomNumber << setw(20) << roomType << endl;  }  fin.close();  }  void searchByLastName(const char\* filename, const char\* searchLastName) {  ifstream fin(filename);  if (!fin) {  cerr << "Ошибка открытия файла!" << endl;  return;  }  HotelGuest guest;  bool found = false;  int i = 0;  while (!fin.fail()) {  i++;  fin >> guest.passportData >> guest.lastName;  if (strcmp(guest.lastName, searchLastName) == 0) {  fin >> guest.firstName >> guest.middleName >> guest.checkInDate >> guest.checkOutDate >> guest.roomNumber >> guest.roomType;  cout << "Гость номер: " << i;  found = true;  }  }  if (!found) {  cout << "Гость не найден." << endl;  }  fin.close();  }  int main() {  setlocale(LC\_ALL, "rus");  const char filename[] = "hotel\_guests.txt";  int choice;  do {  cout << "\n1. Добавить гостя\n";  cout << "2. Вывести список гостей\n";  cout << "3. Поиск по фамилии\n";  cout << "0. Выход\n";  cout << "Введите номер операции: ";  cin >> choice;  switch (choice) {  case 1:  writeToFile(filename);  break;  case 2:  readFromFile(filename);  break;  case 3: {  char searchLastName[50];  cout << "Введите фамилию для поиска: ";  cin >> searchLastName;  searchByLastName(filename, searchLastName);  break;  }  case 0:  break;  default:  cout << "Неверный ввод!" << endl;  }  } while (choice != 0);  return 0;  } |  |

**Вариант 13**

|  |  |
| --- | --- |
| **Преподаватели**. Фамилия преподавателя, название экзамена, дата экзамена. Выбор по дате экзамена. Дату экзамена реализовать с помощью битового поля. | |
| #include <iostream>  #include <iomanip>  #include <cstring>  using namespace std;  struct Date {  unsigned int day : 5;  unsigned int month : 4;  unsigned int year : 12;  };  struct Teacher {  char lastName[50];  char examName[50];  Date examDate;  };  void readTeacher(Teacher teachers[], int& size) {  cout << "Введите фамилию преподавателя: ";  cin.getline(teachers[size].lastName, 50);  cout << "Введите название экзамена: ";  cin.getline(teachers[size].examName, 50);  cout << "Введите дату экзамена (день месяц год): ";  unsigned int d, m, y;  cin >> d >> m >> y;  teachers[size].examDate = { d,m,y };  cin.ignore();  size++;  }  void writeTeachers(Teacher teachers[], int size) {  cout << left << setw(20) << "Фамилия" << setw(30) << "Название экзамена" << setw(15) << "Дата экзамена" << endl;  for (int i = 0; i < size; i++) {  cout << left << setw(20) << teachers[i].lastName  << setw(30) << teachers[i].examName  << setw(2) << teachers[i].examDate.day << '.' << setw(2) << teachers[i].examDate.month << '.' << setw(8) << teachers[i].examDate.year << endl;  }  }  int searchTeacher(Teacher teachers[], int size, Date date) {  for (int i = 0; i < size; i++) {  if (teachers[i].examDate.day == date.day && teachers[i].examDate.month == date.month && teachers[i].examDate.year == date.year) {  return i;  }  }  return -1;  }  int main() {  setlocale(LC\_ALL, "rus");  Teacher teachers[20];  int size = 0, choice;  do {  cout << "\n1. Добавить преподавателя\n";  cout << "2. Вывести список преподавателей\n";  cout << "3. Поиск по дате экзамена\n";  cout << "0. Выход\n";  cout << "Выберите операцию: ";  cin >> choice;  cin.ignore();  switch (choice) {  case 1:  readTeacher(teachers, size);  break;  case 2:  writeTeachers(teachers, size);  break;  case 3: {  cout << "Введите дату экзамена (день месяц год): ";  unsigned int d, m, y;  cin >> d >> m >> y;  Date date = { d,m,y };  int index = searchTeacher(teachers, size, date);  if (index == -1) {  cout << "Преподаватель не найден\n";  }  else {  cout << "Преподаватель: " << teachers[index].lastName << ", экзамен: " << teachers[index].examName << "\n";  }  break;  }  case 0:  break;  default:  cout << "Неверный ввод\n";  break;  }  } while (choice != 0);  return 0;  } |  |
| **Студенты**. Ф.И.О., дата поступления, специальность, группа, факультет, средний балл. Выбор по среднему баллу. | |
| #include <iostream>  #include <fstream>  #include <iomanip>  #include <cstring>  using namespace std;  union Student {  char lastName[50];  char firstName[50];  char middleName[50];  char admissionDate[15];  char specialty[50];  char group[10];  char faculty[50];  double averageGrade;  };  void writeToFile(const char\* filename) {  ofstream fout(filename);  if (!fout) {  cerr << "Ошибка открытия файла!" << endl;  return;  }  Student student;  cout << "Введите фамилию: ";  cin >> student.lastName;  fout << student.lastName << "\t";  cout << "Введите имя: ";  cin >> student.firstName;  fout << student.firstName << "\t";  cout << "Введите отчество: ";  cin >> student.middleName;  fout << student.middleName << "\t";  cout << "Введите дату поступления (ДД.ММ.ГГГГ): ";  cin >> student.admissionDate;  fout << student.admissionDate << "\t";  cout << "Введите специальность: ";  cin >> student.specialty;  fout << student.specialty << "\t";  cout << "Введите группу: ";  cin >> student.group;  fout << student.group << "\t";  cout << "Введите факультет: ";  cin >> student.faculty;  fout << student.faculty << "\t";  cout << "Введите средний балл: ";  cin >> student.averageGrade;  fout << student.averageGrade << '\n';  fout.close();  }  void readFromFile(const char\* filename) {  ifstream fin(filename);  if (!fin) {  cerr << "Ошибка открытия файла!" << endl;  return;  }  Student student;  cout << left << setw(20) << "Фамилия" << setw(20) << "Имя" << setw(20) << "Отчество"  << setw(15) << "Дата поступления" << setw(25) << "Специальность"  << setw(10) << "Группа" << setw(20) << "Факультет" << setw(10) << "Средний балл" << endl;  while (!fin.fail()) {  fin >> student.lastName;  if (fin.fail()) break;  char lastName[50];  strncpy\_s(lastName, student.lastName, sizeof(lastName) - 1);  fin >> student.firstName;  if (fin.fail()) break;  char firstName[50];  strncpy\_s(firstName, student.firstName, sizeof(firstName) - 1);  fin >> student.middleName;  if (fin.fail()) break;  char middleName[50];  strncpy\_s(middleName, student.middleName, sizeof(middleName) - 1);  fin >> student.admissionDate;  if (fin.fail()) break;  char admissionDate[15];  strncpy\_s(admissionDate, student.admissionDate, sizeof(admissionDate) - 1);  fin >> student.specialty;  if (fin.fail()) break;  char specialty[50];  strncpy\_s(specialty, student.specialty, sizeof(specialty) - 1);  fin >> student.group;  if (fin.fail()) break;  char group[10];  strncpy\_s(group, student.group, sizeof(group) - 1);  fin >> student.faculty;  if (fin.fail()) break;  char faculty[50];  strncpy\_s(faculty, student.faculty, sizeof(faculty) - 1);  fin >> student.averageGrade;  if (fin.fail()) break;  cout << left << setw(20) << lastName << setw(20) << firstName << setw(20) << middleName  << setw(15) << admissionDate << setw(25) << specialty << setw(10) << group  << setw(20) << faculty << setw(10) << student.averageGrade << endl;  }  fin.close();  }  void searchByAverageGrade(const char\* filename, double minGrade) {  ifstream fin(filename);  if (!fin) {  cerr << "Ошибка открытия файла!" << endl;  return;  }  Student student;  bool found = false;  int i = 0;  cout << "\nСтуденты со средним баллом выше " << minGrade << ":\n";  cout << left << setw(15) << "Фамилия" << setw(15) << "Имя" << setw(15) << "Отчество"  << setw(15) << "Дата поступления" << setw(20) << "Специальность" << setw(10)  << "Группа" << setw(20) << "Факультет" << setw(10) << "Средний балл" << endl;  while (!fin.fail()) {  fin >> student.lastName >> student.firstName >> student.middleName;  fin >> student.admissionDate >> student.specialty >> student.group >> student.faculty >> student.averageGrade;  if (fin.fail()) break;  if (student.averageGrade >= minGrade) {  cout << left << setw(15) << student.lastName << setw(15) << student.firstName << setw(15) << student.middleName  << setw(15) << student.admissionDate << setw(20) << student.specialty << setw(10)  << student.group << setw(20) << student.faculty << setw(10) << student.averageGrade << endl;  found = true;  }  }  if (!found) {  cout << "Студенты с указанным средним баллом не найдены." << endl;  }  fin.close();  }  int main() {  setlocale(LC\_ALL, "rus");  const char filename[] = "students.txt";  int choice;  do {  cout << "\n1. Добавить студента\n";  cout << "2. Вывести список студентов\n";  cout << "3. Поиск по среднему баллу\n";  cout << "0. Выход\n";  cout << "Введите номер операции: ";  cin >> choice;  switch (choice) {  case 1:  writeToFile(filename);  break;  case 2:  readFromFile(filename);  break;  case 3: {  double minGrade;  cout << "Введите минимальный средний балл для поиска: ";  cin >> minGrade;  searchByAverageGrade(filename, minGrade);  break;  }  case 0:  break;  default:  cout << "Неверный ввод!" << endl;  }  } while (choice != 0);  return 0;  } |  |