Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный технологический университет»

**Лабораторная работа**

На тему:

" Представление информации в виде структуры. Индексирование. Union. Enum «

**Выполнил:**

Волковец Сергей Николаевич

(подпись)

**Научный руководитель:**

Белодед Николай Иванович

(подпись)

## Теоритические сведения:

**Структура** – это пользовательский тип данных, совокупность логически связанных данных различного типа, объединенных под одним идентификатором. В языках С или C++ структура создается при помощи ключевого слова struct, за которым следуют необязательное поле тега и список элементов структуры.

struct поле\_тега {  
 тип\_элемента элемент 1;  
 тип\_элемента элемент 2;  
 ...  
 тип\_элемента\_элемент N;  
}; // точка с запятой обязательны

В C++, есть несколько способов получения доступа к членам структуры:

1. Оператор точки (.) - этот оператор используется для доступа к членам структуры через имя экземпляра структуры. Например:

|  |
| --- |
| 1. #include <iostream> using namespace std;   struct Person {  char firstname[20];  char job[30];  int age; };   int main() {  setlocale(LC\_CTYPE, "rus");  Person john;  strcpy\_s(john.firstname, "Джон");  john.age = 30;  strcpy\_s(john.job, "Программист");  cout << "Имя: " << john.firstname << endl; // Имя: Джон  cout << "Возраст: " << john.age << endl; // Возраст: 30  cout << "Работа: " << john.job << endl; // Работа: Программист 2. return 0; } |

2. Оператор стрелки (->) - этот оператор используется для доступа к членам структуры через указатель на экземпляр структуры. Например:

|  |
| --- |
| #include <iostream> using namespace std;   struct Person {  char firstname[20];  int age;  char job[30]; };   int main() {  setlocale(LC\_CTYPE, "rus");  Person\* john = new Person;  strcpy\_s(john->firstname, "Джон");  john->age = 30;  strcpy\_s(john->job, "Программист");  cout << "Имя: " << john->firstname << endl; // Имя: Джон  cout << "Возраст: " << john->age << endl; // Возраст: 30  cout << "Работа: " << john->job << endl; // Работа: программист  delete john;  return 0; } |

**Кроме того мы можем инициализировать структуру, присвоив ее переменным значения с помощью синтаксиса инициализации:**

|  |  |
| --- | --- |
| **1** | **person tom{ 34, "Tom" };** |

**Массив структуры** - это массив, содержащий элементы типа структуры. То есть, каждый элемент массива является экземпляром структуры. Массивы структур могут быть очень полезными в C++, когда требуется хранить и обрабатывать большое количество данных, каждое из которых состоит из нескольких связанных элементов. Обратимся к предыдущему примеру:

|  |
| --- |
| struct Person {  char firstname[20];  char hobby[30];  int age; }; |

|  |
| --- |
| Person people[3]; |

Здесь мы создаем массив people, содержащий три элемента типа Person. Мы можем обращаться к каждому элементу массива, используя оператор индексации [] и записывать значения его полей. Например:

|  |
| --- |
| strcpy\_s(people[0].firstname, "John"); people[0].age = 30; strcpy\_s(people[0].hobby, "Software Engineer");   strcpy\_s(people[1].firstname, "Kate"); people[0].age = 24; strcpy\_s(people[0].hobby, "Sport"); |

**Объединение** – это группирование переменных, которые разделяют одну и ту же область памяти. В зависимости от интерпретации осуществляется обращение к той или другой переменной объединения.

Объединение позволяет представить в компактном виде данные, которые могут изменяться. Одни и те же данные могут быть представлены разными способами с помощью объединений.  
 Для определения объединений применяется ключевое слово union и следующий синтаксис:

|  |
| --- |
| union имя\_объединения  {  тип элемент1;  тип элемент2;  .............  тип элементN;  }; |

|  |
| --- |
| union ascii //наше объединение {  int digit; //числово код символа  char letter; //сам символ }; |

Рассмотрим объединения на примере. Создадим простейшее объединение, которое хранит символ и его числовой код из таблицы ASCII:

После определения объединения мы можем создать его переменную и присвоить ей какое-либо значение. При определении переменной объединения мы ее можем сразу инициализировать, но стоит учитывать, что инициализировать мы можем только один элемент объединения:

|  |
| --- |
| union ascii code = { 120 } //определение переменной и присвоение ей значения целого числа |

Применительно к объединениям существуют несколько ограничений:  
 1. Размер конечного объединения не может быть больше размера самого большого поля.  
 2. В объединении можно использовать только одно поле за раз.  
 3. Объединение не может находиться в статической памяти.  
 4. Объединение не может иметь конструктор или деструктор.  
 5. Доступ к полям объединения должен быть осуществлен через используемое в данный момент поле.  
 6. Объединение нельзя копировать и передавать его в функцию. Если требуется использование объединения в качестве параметра, лучше передавать ссылку на объединение.

**Перечисление** — это список именованных целочисленных констант. Перечисления используются в тех случаях, когда переменные создаваемого типа могут принимать заранее известное конечное множество значений.  
 Для создания перечисления используется ключевое слово enum. Общая форма перечисления имеет следующий вид:

|  |
| --- |
| enum class имя\_перечисления { константа\_1, константа\_2, ... константа\_N }; |

Создадим перечисление:

|  |
| --- |
| enum Colors {  RED,  GREEN,  BLUE, }; |

Каждой константе сопоставляется некоторое числовое значение. По умолчанию первая константа получает в качестве значения 0, а остальные увеличиваются на единицу.

Каждой константе сопоставляется некоторое числовое значение. По умолчанию первая константа получает в качестве значения 0, а остальные увеличиваются на единицу.

Пример демонстрирующий присвоение значений в перечислении:

|  |
| --- |
| #include <iostream> using namespace std;  enum Colors {  RED, //0  GREEN, //1  BLUE, //2 };  int main()  {  Colors currentColor = BLUE;  cout << "Number of blue color is : " << currentColor << endl;//Number of blue color is : 2  return 0; } |

Мы также можем управлять установкой значений в перечислении. Так, мы можем задать начальное значение для одной константы, тогда у последующих констант значение увеличивается на единицу:

|  |
| --- |
| #include <iostream> using namespace std;  enum Colors {  RED = 3,  GREEN,//4  BLUE,//5 };  int main() {  Colors currentColor = BLUE;  cout << "Number of blue color is : " << currentColor << endl;//Number of blue color is : 5  return 0; } |

|  |
| --- |
| #include <iostream> using namespace std;  enum Colors {  RED = 2,  ORANGE,//3  YELLOW,//4  GREEN,//5  BLUE,//6  VIOLETTE = 1  };  int main() {  Colors currentColor = BLUE;  cout << "Number of blue color is : " << currentColor << endl;//Number of blue color is : 6  return 0; } |

Можно назначить каждой константе индивидуальное значение или сочетать этот подход с автоустановкой:

# Задания:

1) **.** Тема: Доставка еды

Предстоит работа с отслеживанием деятельности сервиса доставки

Реализовать структуры:

1. База меню. В базе содержится информация о названии блюд, цене, кратком описании и составе.

2.База клиентов. Содержится информация о фамилии, имени, его контактный телефон, адрес. Имеется информация потраченных за всё время средств, при достижении определённой суммы клиенту предоставляется скидка 10% на дальнейшие заказы.

1)

#include <iostream>

#include <string>

#include <windows.h>

#include <time.h>

#define size 100000

using namespace std;

struct eat {

string name;

int price;

string shortdess;

string comprouds;

};

const string nameoffood[] = {

"Макароны по-флотски",

"Стейк",

"Хлеб",

"Котлеты",

"Хачапури"

};

const string comproud[] = {

"Рецепт 1",

"Рецепт 2",

"Рецепт 3",

"Рецепт 4",

"Рецепт 5"

};

const string shortdes[] = {

"Не берите",

"Вроде вкусно",

"Ай, дружище, бери, не пожалеешь",

"Ай, брат, просто пальцы будешь облизывать",

"Ай-ай-ай"

};

struct eat menu[100];

int main() {

srand(time(NULL));

setlocale(LC\_ALL, "Russian");

int n = rand() % 100;

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

menu[i].name = nameoffood[rand() % 5];

menu[i].price = rand() % size;

menu[i].shortdess = shortdes[rand() % 5];

menu[i].comprouds = comproud[rand() % 5];

}

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

cout << menu[i].name << " ";

cout << menu[i].price << " ";

cout << menu[i].shortdess << " ";

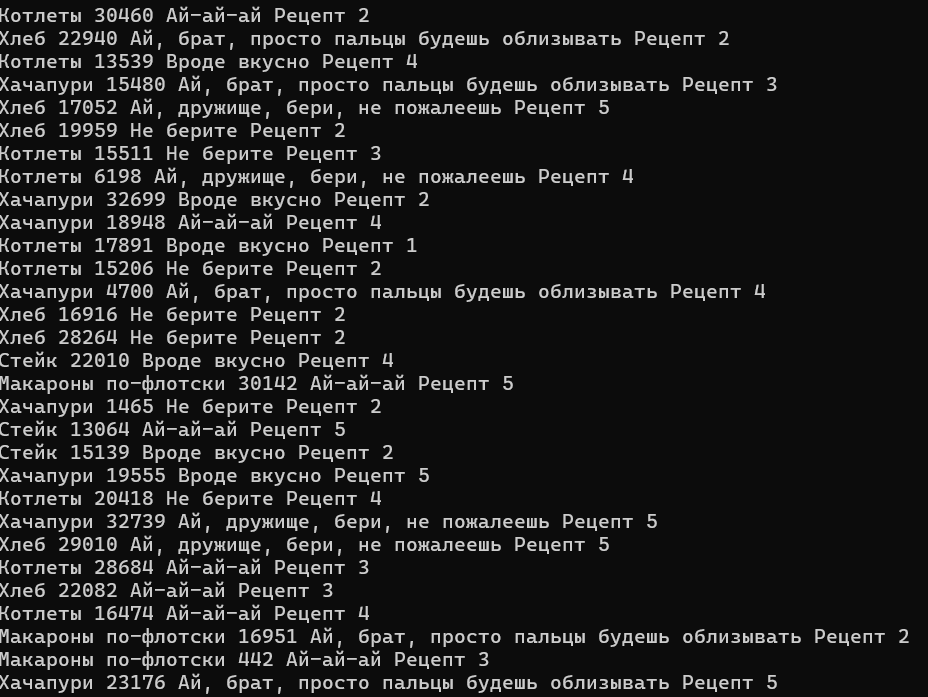
cout << menu[i].comprouds << endl;

}

return 0;

}

Результат работы:



2)

#include <iostream>

#include <string>

#include <time.h>

#include <windows.h>

using namespace std;

struct person

{

string surname;

string name;

string phone;

string address;

int out\_money;

bool diskountt = false;

};

struct person clients[100];

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(NULL));

setlocale(LC\_ALL, "Russian");

const int diskount = 200;

int n = 0;

int choice = 1;

while (choice != 0) {

cout << "1 - ввод элементов структуры с клавиатуры\n";

cout << "2 - вывод элементов структуры в консольное окно\n";

cout << "0 - Выход из программы\n\n";

cout << "Введите номер операции: ";

cin >> choice;

if (choice == 1) {

cout << "Введите данные: Фамилию, Имя,Контактный телефон, адрес" << endl;

cout << "Введите вашу фамилию ";

cin >> clients[n].surname;

cout << "Введите ваше имя ";

cin >> clients[n].name;

cout << "Введите ваш контактный телефон ";

cin >> clients[n].phone;

cout << "Введите ваш адрес ";

cin >> clients[n].address;

clients[n].out\_money = rand() % 1000;

if (clients[n].out\_money >= diskount) {

clients[n].diskountt = true;

cout << "Поздравляем вы получаете скидку в 10% " << endl;

}

n++;

}

if (choice == 2) {

cout << "База данных" << endl;

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

cout << clients[i].surname << " ";

cout << clients[i].name << " ";

cout << clients[i].phone << " ";

cout << clients[i].address << " ";

if (clients[i].diskountt == true) {

cout << "Имеется скидка в десять процентов";

}

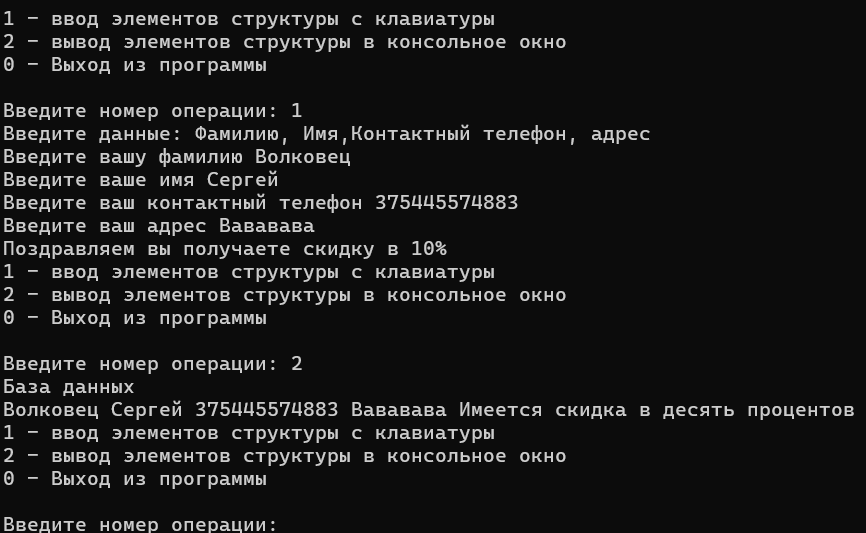
cout << endl;

}

}

}

}



3) Написать программу, в которой нужно создать объединение для хранения двух типов данных (например, целых чисел и логических значений) и написать функцию для вывода этих значений на экран.

#include <iostream>

#include <time.h>

using namespace std;

union two {

int number;

bool b;

};

two many[100];

void out(two many[], int n) {

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

cout << many[i].number<<" ";

cout << many[i].b << endl;

}

}

int main()

{

srand(time(NULL));

int n = rand() % 100;

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

many[i].number = rand() % 1000;

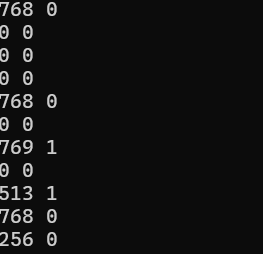
if (many[i].number % 2 != 0) many[i].b = true;

else many[i].b = false;

}

out(many, n);

}

цфй

inf m;

m.ii = 5;

cout << "Представление в виде целого числа: " << m.ii << endl;

cout << "Представление в виде двоичного кода: ";

for (int i = 0; i < sizeof(m.bArray); ++i) {

cout << bitset<8>(m.bArray[i]) << " ";

}

cout << endl;

return 0;

}



5) Написать программу, которая, в зависимости от выбора пользователя, в консоли должна выводить название планеты Солнечной системы, с и #include <iostream>

using namespace std;

enum Planets {

Меркурий = 1,

Венера,

Земля,

Марс,

Юпитер,

Сатурн,

Уран,

Нептун

};

void PlName(int planet) {

switch (planet) {

case Меркурий: cout << "Меркурий" << endl; break;

case Венера: cout << "Венера" << endl; break;

case Земля: cout << "Земля" << endl; break;

case Марс: cout << "Марс" << endl; break;

case Юпитер: cout << "Юпитер" << endl; break;

case Сатурн: cout << "Сатурн" << endl; break;

case Уран: cout << "Уран" << endl; break;

case Нептун: cout << "Нептун" << endl; break;

default: cout << "Неверный выбор" << endl; break;

}

}

int main() {

setlocale(LC\_ALL, "ru");

int choice;

cout << "Выберите номер планеты Солнечной системы 1-8 ";

cin >> choice;

PlName(choice);

return 0;

}

с использованием перечислений.

