

Министерство образования Республики Беларусь
«ПОЛОЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. ЕВФРОСИНИИ
ПОЛОЦКОЙ»

Факультет информационных технологий
Кафедра технологий программирования

**Методические указания для выполнения
лабораторной работы №9
по курсу «Конструирование программного
обеспечения»**

«Работа со структурными типами данных в языке высокого уровня»

Полоцк, 2022 г.

ЦЕЛЬ РАБОТЫ

Познакомится с таким понятием как структура. Разобрать методы для работы со структурами в C++ и C#. На основе примеров, приведенных в данной лабораторной работе, выполнить свой вариант практического задания.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Структуры в C#

Наряду с классами структуры представляют еще один способ создания собственных типов данных в C#. Более того многие примитивные типы, например, `int`, `double` и т.д., по сути являются структурами.

Определение структуры

Для определения структуры применяется ключевое слово `struct`:

```
struct имя_структуры
{
    // элементы структуры
}
```

После слова `struct` идет название структуры и далее в фигурных скобках размещаются элементы структуры - поля, методы и т.д.

Например, определим структуру, которая будет называться `Person` и которая будет представлять человека:

```
new String('a', 6)
```

Как и классы, структуры могут хранить состояние в виде полей (переменных) и определять поведение в виде методов. Например, добавим в структуру `Person` пару полей и метод:

```
struct Person
{
    public string name;
    public int age;

    public void Print()
    {
        Console.WriteLine($"Имя: {name}  Возраст: {age}");
    }
}
```

В данном случае определены две переменные - name и age для хранения соответственно имени и возраста человека и метод Print для вывода информации о человеке на консоль.

И, как и в случае с классами, для обращения к функциональности структуры - полям, методам и другим компонентам структуры применяется точечная нотация - после объекта структуры ставится точка, а затем указывается компонент структуры:

```
объект.поле_структуры  
объект.метод_структуры(параметры_метода)
```

Создание объекта структуры

Инициализация с помощью конструктора

Для использования структуры ее необходимо инициализировать. Для инициализации создания объектов структуры, как и в случае с классами, применяется вызов конструктора с оператором new. Даже если в коде структуры не определено ни одного конструктора, тем не менее имеет как минимум один конструктор - конструктор по умолчанию, который генерируется компилятором. Этот конструктор не принимает параметров и создает объект структуры со значениями по умолчанию.

```
new название_структуры();
```

Например, создадим объект структуры Person с помощью конструктора по умолчанию:

```
Person tom = new Person(); // вызов конструктора  
// или так  
// Person tom = new();  
  
tom.name = "Tom"; // изменяем значение по умолчанию в поле name  
  
tom.Print(); // Имя: Tom Возраст: 0  
  
struct Person  
{  
    public string name;  
    public int age;  
  
    public void Print()  
    {  
        Console.WriteLine($"Имя: {name} Возраст: {age}");  
    }  
}
```

В данном случае создается объект `tom`. Для его создания вызывается конструктор по умолчанию, который устанавливает значения по умолчанию для его полей. Для числовых данных это значение 0, поэтому поле `age` будет иметь значение 0. Для строк это значение `null`, которое указывает на отсутствие значения. Но далее, если поля доступны (а в данном случае поскольку они имеют модификатор `public` они доступны), мы можем изменить их значения. Так, здесь полю `name` присваивается строка «Том». Соответственно при выполнении метода `Print()` мы получим следующий консольный вывод:

```
Имя: Tom  Возраст: 0
```

Непосредственная инициализация полей

Если все поля структуры доступны (как в случае с полями структуры `Person`, который имеет модификатор `public`), то структуру можно инициализировать без вызова конструктора. В этом случае необходимо присвоить значения всем полям структуры перед получением значений полей и обращением к методам структуры. Например:

```
Person tom;           // не вызываем конструктор

// инициализация полей
tom.name = "Sam";
tom.age = 37;

tom.Print();          // Имя: Sam  Возраст: 37

struct Person
{
    public string name;
    public int age;

    public void Print()
    {
        Console.WriteLine($"Имя: {name}  Возраст: {age}");
    }
}
```

Инициализация полей по умолчанию

Стоит отметить, что начиная с версии C# 10, мы можем напрямую инициализировать поля структуры при их определении (до C# 10 это делать было нельзя):

```

Person tom = new Person();
tom.Print();      // Имя:Том  Возраст: 1

struct Person
{
    // инициализация полей значениями по умолчанию - доступна с
    C#10
    public string name = "Tom";
    public int age = 1;
    public Person() { }
    public void Print() => Console.WriteLine($"Имя:
{name}  Возраст: {age}");
}

```

Однако даже в этом случае, несмотря на значения по умолчанию, необходимо явно определить и вызывать конструктор, если мы хотим использовать эти значения.

Конструкторы структуры

Как и класс, структура может определять конструкторы. Однако, если в структуре определяется конструктор, то в нем обязательно надо инициализировать все поля структуры.

Например, добавим в структуру Person конструктор:

```

Person tom = new();
Person bob = new("Bob");
Person sam = new("Sam", 25);

tom.Print();      // !!!! Имя:  Возраст: 0
bob.Print();      // Имя: Bob  Возраст: 1
sam.Print();      // Имя: Sam  Возраст: 25

struct Person
{
    public string name;
    public int age;

    public Person(string name = "Tom", int age = 1)
    {
        this.name = name;
        this.age = age;
    }
    public void Print() => Console.WriteLine($"Имя:
{name}  Возраст: {age}");
}

```

В данном случае в структуре `Person` определен конструктор с двумя параметрами, для которых предоставлены значения по умолчанию. Однако обратите внимание на создание первого объекта структуры:

```
Person tom = new(); // по прежнему используется конструктор без
параметров по умолчанию
tom.Print();        // !!!! Имя:    Возраст: 0
```

Здесь по-прежнему применяется конструктор по умолчанию, тогда как при инициализации остальных двух переменных структуры применяется явно определенный конструктор.

Однако начиная с версии C#10 мы можем определить свой конструктор без параметров:

```
Person tom = new();

tom.Print();        // Имя: Том  Возраст: 37

struct Person
{
    public string name;
    public int age;

    public Person()
    {
        name = "Tom";
        age = 37;
    }

    public void Print() => Console.WriteLine($"Имя:
{name}  Возраст: {age}");
}
```

Опять же при определении конструктора без параметров необходимо инициализировать все поля структуры.

В случае если нам необходимо вызывать конструкторы с различным количеством параметров, то мы можем, как и в случае с классами, вызывать их по цепочке:

```
Person tom = new();
Person bob = new("Bob");
Person sam = new("Sam", 25);

tom.Print();        // Имя: Том  Возраст: 1
bob.Print();        // Имя: Bob  Возраст: 1
sam.Print();        // Имя: Sam  Возраст: 25
```

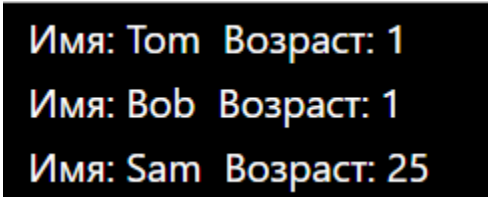
```

struct Person
{
    public string name;
    public int age;

    public Person() : this("Tom")
    { }
    public Person(string name) : this(name, 1)
    { }
    public Person(string name, int age)
    {
        this.name = name;
        this.age = age;
    }
    public void Print() => Console.WriteLine($"Имя:
{name}  Возраст: {age}");
}

```

Конструкторы по-прежнему должны инициализировать значения всех полей, однако поскольку при вызове любого конструктора цепочка все равно закончится на последнем конструкторе, который выполняет инициализацию, то инициализацию полей в других конструкторах можно не делать. Консольный вывод программы:



```

Имя: Tom  Возраст: 1
Имя: Bob  Возраст: 1
Имя: Sam  Возраст: 25

```

Инициализатор структуры

Также, как и для класса, можно использовать инициализатор для создания структуры:

```

Person tom = new Person { name = "Tom", age = 22 };

tom.Print();    // Имя: Tom  Возраст: 22

struct Person
{
    public string name;
    public int age;
    public void Print() => Console.WriteLine($"Имя:
{name}  Возраст: {age}");
}

```

При использовании инициализатора сначала вызывается конструктор без параметров: если мы явным образом не определили конструктор без параметров, то вызывается конструктор по умолчанию. А затем его полям присваиваются соответствующие значения.

Копирование структуры с помощью with

Если нам необходимо скопировать в один объект структуры значения из другого с небольшими изменениями, то мы можем использовать оператор `with`:

```
Person tom = new Person { name = "Tom", age = 22 };
Person bob = tom with { name = "Bob" };
bob.Print();    // Имя: Bob  Возраст: 22
```

В данном случае объект `bob` получает все значения объекта `tom`, а затем после оператора `with` в фигурных скобках указывается поля со значениями, которые мы хотим изменить.

Структуры в C++

Структура в языке C++ представляет собой производный тип данных, который представляет какую-то определенную сущность, также, как и класс. Нередко структуры применительно к C++ также называют классами. И в реальности различия между ними не такие большие.

Для определения структуры применяется ключевое слово `struct`, а сам формат определения выглядит следующим образом:

```
struct имя_структуры
{
    компоненты_структуры
};
```

`Имя_структуры` представляет произвольный идентификатор, к которому применяются те же правила, что и при наименовании переменных.

После имени структуры в фигурных скобках помещаются `Компоненты_структуры`, которые представляют набор описаний объектов и функций, которые составляют структуру.

Например, определим простейшую структуру:

```
#include <iostream>
#include <string>
```



```

struct person
{
    int age;
    std::string name;
};

int main()
{
    person tom;
    tom.name = "Tom";
    tom.age = 34;
    std::cout << "Name: " << tom.name << "\tAge: " << tom.age <<
std::endl;
    return 0;
}

```

Здесь определена структура `person`, которая имеет два элемента: `age` (представляет тип `int`) и `name` (представляет тип `string`).

После определения структуры мы можем ее использовать. Для начала мы можем определить объект структуры - по сути обычную переменную, которая будет представлять выше созданный тип. Также после создания переменной структуры можно обращаться к ее элементам - получать их значения или, наоборот, присваивать им новые значения. Для обращения к элементам структуры используется операция "точка":

```
имя_переменной_структуры.имя_элемента
```

По сути структура похожа на класс, то есть с помощью структур также можно определять сущности для использования в программе. В то же время все члены структуры, для которых не используется спецификатор доступа (`public`, `private`), по умолчанию являются открытыми (`public`). Тогда как в классе все его члены, для которых не указан спецификатор доступа, являются закрытыми (`private`).

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

```
person tom = { 34, "Tom" };
```

Инициализация структур аналогична инициализации массивов: в фигурных скобках передаются значения для элементов структуры по порядку. Так как в структуре `person` первым определено свойство, которое представляет тип `int` - число, то в фигурных скобках вначале идет число. И так далее для всех элементов структуры по порядку.

При этом любой класс мы можем представить в виде структуры и наоборот. Возьмем, к примеру, следующий класс:

```

class Person
{
public:
    Person(std::string n, int a)
    {
        name = n; age = a;
    }
    void move()
    {
        std::cout << name << " is moving" << std::endl;
    }
    void setAge(int a)
    {
        if (a > 0 && a < 100) age = a;
    }
    std::string getName()
    {
        return name;
    }
    int getAge()
    {
        return age;
    }
private:
    std::string name;
    int age;

};

```

Данный класс определяет сущность человека и содержит ряд приватных и публичных переменных и функций. Вместо класса для определения той же сущности мы могли бы использовать структуру:

```

#include <iostream>
#include <string>

struct user
{
public:
    user(std::string n, int a)
    {
        name = n; age = a;
    }
    void move()
    {
        std::cout << name << " is moving" << std::endl;
    }
}

```

```

    void setAge(int a)
    {
        if (a > 0 && a < 100) age = a;
    }
    std::string getName()
    {
        return name;
    }
    int getAge()
    {
        return age;
    }
private:
    std::string name;
    int age;

};

int main()
{
    user tom("Tom", 22);
    std::cout << "Name: " << tom.getName() << "\tAge: " <<
tom.getAge() << std::endl;
    tom.setAge(31);
    std::cout << "Name: " << tom.getName() << "\tAge: " <<
tom.getAge() << std::endl;
    return 0;
}

```

И в плане конечного результата программы мы не увидели бы никакой разницы.

Ключевое слово `typedef`

Существует более простой способ определения структур, или вы можете создавать типы псевдонимов. Например:

```

typedef struct {
    char   title[50];
    char   author[50];
    char   subject[100];
    int    book_id;
} Books;

```

Теперь вы можете напрямую использовать книги для определения переменных типа `Books` без использования ключевого слова `struct`. Ниже приведен пример:

Books Book1, Book2;

Когда использовать структуры? Как правило, структуры используются для описания таких данных, которые имеют только набор публичных атрибутов - открытых переменных. Например, как та же структура `person`, которая была определена в начале. Иногда подобные сущности еще называют агрегатными классами (aggregate classes).

Содержание отчета

Отчет должен включать:

- а) титульный лист;
- б) формулировку цели работы;
- в) описание результатов выполнения заданий:
 - листинги программ;
 - результаты выполнения программ;
- г) выводы, согласованные с целью работы.

Задания

Вариант 1

1. Описать структуру с именем `STUDENT`, содержащую следующие поля:
 - `NAME` – фамилия и инициалы;
 - `GROUP` – номер группы;
 - `SES` – успеваемость (массив из пяти элементов).
2. Написать программу, выполняющую следующие действия:
 - ввод с клавиатуры данных в массив `STUD1`, состоящий из десяти структур типа `STUDENT`; записи должны быть упорядочены по возрастанию среднего балла;
 - вывод на дисплей фамилий и номеров групп для всех студентов, имеющих оценки 4 и 5;
 - если таких нет, вывести соответствующее сообщение.

Вариант 2

1. Описать структуру с именем `STUDENT`, содержащую следующие поля:
 - `NAME` – фамилия и инициалы;
 - `GROUP` – номер группы;
 - `SES` – успеваемость (массив из пяти элементов).
2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив STUD1, состоящий из десяти структур типа STUDENT; записи должны быть упорядочены по алфавиту;
- вывод на дисплей фамилий и номеров групп для всех студентов, имеющих хотя бы одну оценку 2;
- если таких студентов нет, вывести соответствующее сообщение.

Вариант 3

1. Описать структуру с именем AEROFLOT, содержащую следующие поля:
 - NAZN – название пункта назначения рейса;
 - NUMR – номер рейса;
 - TIP – тип самолета.
2. Написать программу, выполняющую следующие действия:
 - ввод с клавиатуры данных в массив AIRPORT, состоящий из семи элементов типа AEROFLOT; записи должны быть упорядочены по возрастанию номера рейса;
 - вывод на экран номеров рейсов и типов самолетов, вылетающих в пункт назначения, название которого совпало с названием, введенным с клавиатуры;
 - если таких рейсов нет, выдать на дисплей соответствующее сообщение.

Вариант 4

1. Описать структуру с именем WORKER, содержащую следующие поля:
 - NAME – фамилия и инициалы работника;
 - POS – название занимаемой должности;
 - YEAR – год поступления на работу.
2. Написать программу, выполняющую следующие действия:
 - ввод с клавиатуры данных в массив TABL, состоящий из десяти структур типа WORKER; записи должны быть размещены по алфавиту.
 - вывод на дисплей фамилий работников, чей стаж работы в организации превышает значение, введенное с клавиатуры;
 - если таких работников нет, вывести на дисплей соответствующее сообщение.

Вариант 5

1. Описать структуру с именем TRAIN, содержащую следующие поля:
 - NAZN – название пункта назначения;
 - NUMR – номер поезда;
 - TIME – время отправления.
2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив RASP, состоящий из восьми элементов типа TRAIN; записи должны быть упорядочены по номерам поездов;
- вывод на экран информации о поезде, номер которого введен с клавиатуры;
- если таких поездов нет, выдать на дисплей соответствующее сообщение.

Вариант 6

1. Описать структуру с именем MARSH, содержащую следующие поля:
 - BEGST – название начального пункта маршрута;
 - TERM – название конечного пункта маршрута;
 - NUMER – номер маршрута.
2. Написать программу, выполняющую следующие действия:
 - ввод с клавиатуры данных в массив TRAFIC, состоящий из восьми элементов типа MARSH; записи должны быть упорядочены по номерам маршрутов;
 - вывод на экран информации о маршрутах, которые начинаются или кончаются в пункте, название которого введено с клавиатуры;
 - если таких маршрутов нет, выдать на дисплей соответствующее сообщение.

Вариант 7

1. Описать структуру с именем NOTE, содержащую следующие поля:
 - NAME – фамилия, имя;
 - TELE – номер телефона;
 - BDAY – день рождения (массив из трех чисел).
2. Написать программу, выполняющую следующие действия:
 - ввод с клавиатуры данных в массив BLOCKNOTE, состоящий из восьми элементов типа NOTE; записи должны быть упорядочены по датам дней рождения;
 - вывод на экран информации о человеке, номер телефона которого введен с клавиатуры;
 - если такого нет, выдать на дисплей соответствующее сообщение.

Вариант 8

1. Описать структуру с именем ZNAK, содержащую следующие поля:
 - NAME – фамилия, имя;
 - ZODIAC – знак Зодиака;
 - BDAY – день рождения (массив из трех чисел).
2. Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив BOOK, состоящий из восьми элементов типа ZNAK; записи должны быть упорядочены по датам дней рождения;
- вывод на экран информации о человеке, чья фамилия введена с клавиатуры;
- если такого нет, выдать на дисплей соответствующее сообщение.

Вариант 9

1. Описать структуру с именем PRICE, содержащую следующие поля:
 - TOVAR – название товара;
 - MAG – название магазина, в котором продается товар;
 - STOIM – стоимость товара в руб.
2. Написать программу, выполняющую следующие действия:
 - ввод с клавиатуры данных в массив SPISOK, состоящий из восьми элементов типа PRICE; записи должны быть размещены в алфавитном порядке по названиям товаров;
 - вывод на экран информации о товаре, название которого введено с клавиатуры;
 - если таких товаров нет, выдать на дисплей соответствующее сообщение.

Вариант 10

1. Описать структуру с именем ORDER, содержащую следующие поля:
 - PLAT – расчетный счет плательщика;
 - POL – расчетный счет получателя;
 - SUMMA – перечисляемая сумма в руб.
2. Написать программу, выполняющую следующие действия:
 - ввод с клавиатуры данных в массив SPISOK, состоящий из восьми элементов типа ORDER; записи должны быть размещены в алфавитном порядке по расчетным счетам плательщиков;
 - вывод на экран информации о сумме, снятой с расчетного счета плательщика, введенного с клавиатуры;
 - если такого расчетного счета нет, выдать на дисплей соответствующее сообщение.

Контрольные вопросы

1. Что такое структура в языке? Назначение структур.
2. Какая общая форма объявления структуры (структурного типа)?
3. Пример объявления структуры, которая описывает информацию о студенте
4. Пример объявления, инициализации и использования структурной переменной
5. Какой тип доступа по умолчанию имеют поля структурной переменной?