**Практическое задание 15**

**Структура как тип данных. Описание типа и объявление объектов структурного типа. Практическое использование структур в прикладных задачах.**

Цель. Изучение структур как типа данных. Реализация алгоритмов работы с данными структурного типа и массивами структур на основе функций. Обмен данными с текстовыми файлами.

**ПЛАН**

[1. Краткое теоретическое введение 1](#_Toc27340820)

[2. Структуры, массивы и указатели 3](#_Toc27340821)

[3. Использование функций для работы со структурами 3](#_Toc27340822)

[4. Обмен данными с текстовыми файлами 4](#_Toc27340823)

[5. Задание 4](#_Toc27340824)

[Выводы 4](#_Toc27340825)

1. Краткое теоретическое введение

**Определение**. Структура, это тип, позволяющий объединить в логическое целое множество данных разного типа.

Назначение − объектное отображение типов данных. Поля (элементы) структуры имеют различные имена и могут быть разных типов.

**Описание структурного типа** конструирует новый тип данного и имеет синтаксис:

struct имя\_типа

{

// определения элементов;

}; // ; завершает описание

struct − ключевое слово, обозначающее структурный тип;

имя\_типа − произвольное имя, которое идентифицирует новый тип;

определения элементов (полей) − описания объектов, входящих в состав структуры, каждый из которых есть прототип одного из данных, входящих в состав описываемого структурного типа.

Пример структуры − товар на складе:

struct Goods {

string name; // Наименование

float price; // Оптовая цена

float percent; // Торговая наценка в %

int vol; // О6ъем партии товара

string date; // Дата поставки партии товара

};

Имя типа – struct Goods. Является именем типа данного, который описывает объект в общем виде. Наименование товара связано с именем name, оптовая цена единицы товара есть значение типа float с названием price, и так далее.

**Объявление объекта** структурного типа.

Введенное описание структурного типа именует абстрактный тип данного, который может использоваться для объявления переменных этого типа, так же, как данных базовых типов. Синтаксис объявления:

struct имя\_структурного\_типа имя\_переменной; // переменная

struct имя\_структурного\_типа \* имя\_переменной; // указатель

Например,

struct Goods food;

struct Goods \*p\_food;

**Примечание**

Для конструирования структур лучше использовать служебное слово typedef, которое позволяет ввести собственное обозначение для любого определения типа. Для структурного типа синтаксис таков:

typedef struct

{

// определения элементов;

} Имя\_типа // Обозначение = имя типа;

Пример для того же объекта:

typedef struct {

string name; // наименование

float price; // оОптовая цена

float percent; // торговая наценка в %

int vol; // о6ъем партии товара

string date; // дата поставки партии товара

} **Goods**;

Это описание вводит структурный тип и присваивает ему имя Goods, которое используется для объявления объектов структурного типа, например:

**Goods** food1, food2, food3;

Объявлены три структуры: food1, food2, food3, каждая из которых представляет реальный объект.

**Выделение памяти и инициализация структур**

Описание структурного типа не связано с выделением памяти. Только при объявлении объекта ему выделяется память в таком количестве, чтобы могли разместиться данные всех элементов структуры.

Инициализация структур похожа на инициализацию массивов. При объявлении объекта структуры в фигурных скобках после имени и знака присваивания размещается список начальных значений элементов, например:

**Goods** coat = {

" Черная шляпа", 1000.00, 15.0, 100, "12.01.04"

};

При этом элементы объекта coat получают соответствующие начальные значения.

**Операции** над структурами

1. Разрешается **присваивание** для структур, если операнды одного типа.

**Goods** dress;

Допустимо следующее присваивание:

dress = coat;

2. Операции **сравнения** для структур не определены, сравнивать структуры можно только по отдельным полям.

3. **Доступ** к элементам структур осуществляет операция разыменования «точка», ее синтаксис:

имя\_структуры.имя\_элемента

Здесь:

имя\_структуры − имя объекта структурного типа,

имя\_элемента − имя одного из элементов в соответствии с определением.

Так, coat.name − строка "пиджак черный".

Перед точкой стоит имя конкретной структуры, для которой при ее объявлении выделена память.

2. Структуры, массивы и указатели

2.1. Массивы и структуры как элементы структур

Массив, как и любые другие данные, может быть элементом структуры, тогда для обращения к каждому элементу такого массива требуется операция разыменования элемента массива.

Элементом структуры может быть другая структура, это называется вложение структур. Например, структурный тип для представления сведений о студенте: имя, фамилия, группа, может содержать данное «дата рождения». Это может быть структура с элементами «число», «месяц», «год».

2.2 Массивы структур.

Массивы структур вводятся в употребление с помощью описания так же, как и массивы других типов данных с указанием имени структурного типа, например:

struct Goods List [MAX];

Определен массив List, состоящий из MAX элементов, каждый из которых является структурой типа Goods. Имя List, это имя массива, элементами которого являются структуры List[0], List[l] и так далее, все они структуры типа Goods.

Для доступа к полям структур, входящих в массив структур, используются уточненные имена с индексированием первого имени (имени массива). Индекс записывается непосредственно после имени массива структур, тем самым из массива выделяется нужная структура, а уже с помощью точки и последующего имени идентифицируется соответствующий компонент структуры, например:

List[0].price, это элемент с именем price структуры типа Goods, входящей в качестве первого элемента (с нулевым индексом) в массив структур List[].

При размещении в памяти массива структур элементы массива размещаются подряд в порядке возрастания индекса. Каждый элемент массива занимает столько места, сколько необходимо для размещения структуры.

Как и массивы других типов, массив структур при определении может быть инициализирован. Инициализатор массива структур должен содержать в фигурных скобках список начальных значений структур массива. В свою очередь, каждое начальное значение для структуры, это список значений ее компонентов (также в фигурных скобках).

3. Использование функций для работы со структурами

Возможны всего два варианта: структура или массив структур может быть передана функции как параметр, или структура (массив) может быть возвращаемым функцией значением. И в том, и в другом случае могут использоваться указатели на объекты структурных типов. Рассмотрим возможные варианты. Пусть есть структура:

struct Person {

string name;

int age;

};

Обязательные синтаксические конструкции в прототипах функций, которые работают со структурами:

1) параметр функции − объект структурного типа:

void F1 (struct Person Имя\_объекта);

2) параметр функции − адрес объекта структурного типа:

void F2 (struct Person &Имя\_объекта);

3) параметр функции − массив объектов структурного типа:

void F3 (struct Person Имя\_массива[], int Len);

4) функция возвращает структуру:

struct Person F4 (Список\_параметров);

5) функция возвращает указатель на структуру:

struct Person \* F5 (Список\_параметров);

Как следует из примеров, принципы передачи параметров в функцию и возвращения значений остаются прежними и неизмеными.

4. Обмен данными с текстовыми файлами

Обмен данными с текстовыми файлами выполняется так же, как для данных любых типов перегазначением консольного ввода/вывода на объекты fstream.

5. Задание

В заголовочном файле My\_struct.h описана структура Person и некоторые функции для работы с одним данным структурного типа и массивом структур. Скопируйте заголовочный файл в папку проекта и присоедините к проекту, откройте и ознакомьтесь с содержимым.

В функции main опишите массив Person All\_Person[20];

Введите переменную для обозначения длины массива.

**Упражнение 1. Обращение к функциям работы с записями**

Введите 2-3 записи, выведите на экран. Обратитесь к функции, вычисляющей средний показатель по одному предмету.

**Упражнение 2. Алгоритм добавления одной записи в массив**

Напишите функцию добавления одной записи в массив с прототипом:

void Add(Person All\_Person[], int &len, Person New);

Функция должна приписать новую запись в конец массива записей.

**Упражнение 3. Запись данных в файл**

Напишите функцию вывода массива записей в файл.

Выводы

1. Когда имеешь дело с табличным представлением данных, логичнее всего использовать массив записей, где одна запись – данное структурного типа.

2. Функции, работающие с записями, получают входные данные и возвращают результат так же точно, как и с данными других типов.

3. main описывает входные данные, и передает управление в функции, реализуя сценарий работы приложения.