|  |
| --- |
| 1. Структуры: установка структурного шаблона, определение структурных переменных, инициализация структуры. Доступ к элементам структуры.   Рассмотрим новый тип данных - структуру. Он не только гибок для представления разнообразных данных, но и позволяет создавать новые (пользовательские) типы данных. Пример использования - создание каталога книг. Каждая книга имеет следующие атрибуты: шифр, название, автора, издательство, год издания, число страниц, тираж, цену. Это несколько массивов. Очень сложно организовать одновременную работу с каталогом, если нужно их упорядочить по названиям, авторам, цене и так далее. Лучше иметь один массив, в котором каждый элемент содержит всю информацию о книге.  **Структура** – это объект, состоящий из последовательностей поименованных  элементов. Каждый элемент имеет свой тип. Для определения нового типа данных нужно его описать (задать шаблон структуры):  *struct book {*  *char title [81];*  *char author[30];*  *float value;*  *};*  *book* – это имя нового типа данных.  **Структурный шаблон** является основной схемой, описывающей, как образуется новый тип. *struct* - ключевое слово, имя типа структуры *book* - необязателен, если сразу определить имя переменной, то его можно не вводить.  *struct {*  *char title [81];*  *char author[30];*  *float value;*  *}libry;*  Каждый элемент структуры определяется своим собственным описанием. Это  переменные и массивы стандартных типов данных. Шаблон является схемой без содержания. Он сообщает компилятору, как сделать что-то, но ничего не делает в программе, а вот создание структурной переменной, это и есть смысл слова «структура». Согласно шаблону под эту переменную выделяется память, равная сумме всех элементов (81).  *struct book* играет ту же роль, что и int, float перед именем переменной.  (struct book doyle, panshin;)  Для доступа к элементам структурной переменной используется операция точка. Имя переменной, точка, имя элемента структуры.  *void main(void) {*  *struct book libry; //описание перем-й типа book*  *puts("Введите название книги");*  *gets(libry.title);*  *puts("Введите фамилию автора"); gets(libry.author);*  *puts("Введите цену книги");*  *scanf("%f",&libry.value);*  *printf("%s, %s, %p.2f",libry.title,libry.author,libry.value);*  *}*  Структурную переменную можно инициализировать:  *struct book libry* ={"Руслан и Людмила", "А.С.Пушкин", 1.50}; |
| 1. Массивы структур: описание, определение элементов массива структур.   Если переменных типа структура много, то определяется массив структур.  *void main(void){*  *struct book libry[100];*  *int i;*  *for(i=0; i<100; i++){*  *puts("Введите название книги");*  *gets(libry[i].title);*  *puts("Введите автора книги");*  *gets(libry[i], author);*  *puts("Введите цену книги");*  *scanf("%f",&libry[i].value);*  *}*  *}*  Индекс применяется к имени массива структур libry[i].  Если libry[2].title[3] – это 4-й элемент в title в 3-й структуре типа book. |
| 1. Вложенные структуры. |
| 1. Битовые структуры. |
| 1. Адресация в языке программирования. Указатели. Операции с указателями. |
| 1. Массивы и указатели. |
| 1. Структуры и указатели. |
| 1. Динамическая память(функции и операции). |
| 1. Списки. |
| 1. Объединения. Преобразование массивов. |
| 1. Типы функций и их определение. Описание параметров. Аргументы функции: формальные и фактические. Наличие нескольких аргументов. Вызов функции. Обращение к функции до ее определения. Возвращение значения функции. |
| 1. Функции. Способы передачи информации в функции. |
| 1. Функции: передача аргументов (числовых, символьных, строк символов) в функции |
| 1. Файлы. Установка файловой переменной. Функции, работающие с файлами: FOPEN(), FCLOSE(), FREAD(). |
| 1. Понятие класса и его свойства: абстрагирование, абстракция, ограничение доступа, иерархия, типизация, параллелизм, устойчивость. |
| 1. Типы отношений между классами. |
| 1. Понятие объекта и его элементы: состояние, поведение, индивидуальность. |
| 1. Отношения между объектами. |
| 1. Взаимоотношения между классами и объектами. |
| 1. Приватные, общедоступные и защищенные члены класса. |
| 1. Конструктор и деструктор класса: объявление, описание и использование. |
| 1. Конструктор с аргументами. |
| 1. Объявление, описание, и использование функций-членов классов. |
| 1. Статические члены-данные. Приватные статические члены-данные. |
| 1. Приватный конструктор и деструктор. |
| 1. Объекты класса как члены-данные. |
| 1. Статические функции-члены: объявление, описание и использование. |
| 1. Указатели на члены-данные класса. Указатели на члены-данные объекта. |
| 1. Конструкторы классов с подобъектами. |
| 1. Массивы объектов классов. |
| 1. Массив указателей на объекты классов. |
| 1. Массивы указатели на члены-данные классов. |
| 1. Массивы указатели на члены-функции классов. |
| 1. Массивы членов-данных классов. |
| 1. Указатели объектов как члены-данные. |
| 1. Перегрузка. Перегруженные функции как члены так и не члены классов. |
| 1. Перегрузка конструкторов. |
| 1. Наследование. Спецификаторы доступа базовых классов. |
| 1. Перегруженные функции в иерархии классов. Разрешение видимости. Сопоставление аргументов. |
| 1. Правило вызова функций базовых классов. Оператор разрешения видимости при множественном наследовании. |
| 1. Переопределение функций |
| 1. Преобразование типов при простом и множественном наследовании. |
| 1. Понятие указателя this. |
| 1. Перегрузка операторов. |
| 1. Перегруженные операторы как функции-члены класса. |
| 1. Аргументы передаваемые в базовый класс. |
| 1. Перегруженный оператор индексирования. |
| 1. Друзья классов: определение, свойства друзей, дружественная функция. |
| 1. Виртуальные базовые классы. Виртуальные и не виртуальные базовые классы вместе. |
| 1. Механизм полиморфизма: виртуальные и не виртуальные функции. |
| 1. Полиморфизм: виртуальные функции. |
| 1. Полиморфизм. Раннее и позднее связывание переопределение виртуальных функций. |
| 1. Перегрузка виртуальных функций. |
| 1. Абстрактные классы. |
| 1. Обработка исключительных ситуаций блок TRY-CATCH |