Lesson\_12

|  |  |
| --- | --- |
| Коллеги, здравствуйте. Рад вас приветствовать на 12м уроке Основы Языка С. На этом занятии мы поговорим о структурах данных в языке С.  Несмотря на то что язык Си создавался в незапамятные времена, уже тогда программисты понимали, что примитивных типов данных недостаточно для комфортного программирования.  Мир вокруг можно моделировать различными способами. Самым естественным из них является представление о нём, как о наборе объектов. У каждого объекта есть свои свойства. Например, для человека это возраст, пол, рост, вес и т.д. Для велосипеда – тип, размер колёс, вес, материал, изготовитель и пр. Для товара в магазине – идентификационный номер, название, группа, вес, цена, скидка и т.д. У классов объектов набор этих свойств одинаковый: все собаки могут быть описаны, с той или иной точностью, одинаковым набором свойств, но значения этих свойств будут разные. Все самолёты обладают набором общих свойств в пределах одного класса. Если же нам надо более точное описание, то можно выделить подклассы: самолёт амфибии, боевые истребители, пассажирские лайнеры – и в пределах уже этих классов описывать объекты. |  |
| Сразу небольшое отступление, для тех кто изучал высокоуровневые языки, такие как Java или С#, в Си отсутствуют классы в том виде в котором вы привыкли их видеть. Так вот, для работы с такими объектом нам необходима конструкция, которая бы могла агрегировать различные типы данных под одним именем – так появились структуры. Т.е. структура данных - это такая сущность, которая объединяет в себе несколько примитивов. Для примера, создадим такую структуру, как простая дробь. В программировании существуют дробные числа и представлены они типами float и double. Но это десятичные дроби. Мы же будем описывать обычную дробь. |  |
| Для описания структуры используется ключевое слово struct и название структуры. Далее в фигурных скобках описываются переменные, входящие в структуру. В нашем примере это будут числитель и знаменатель. Также опишем переменную, которая будет хранить целую часть. У этих переменных не гарантируются инициализационные значения, т.е. мы ничего не присваиваем им изначально, это просто описание, которое говорит компилятору о том, что когда в коде встретится инициализация нашей структуры, для её хранения понадобится вот столько памяти, которую нужно разметить для хранения вот этих переменных. | #include <stdio.h>  struct fraction {  int integer;  int divisible;  int divisor;  }; |
| Для сокращения записи опишем новый тип данных, назовём его ДРОБЬ. Это делается при помощи ключевого слова typedef. Его синтаксис прост, пишем typedef название старого типа данных название нового типа, т.е. как мы будем называть его в дальнейшем. | typedef struct fraction Fraction; |
| Доступ к переменным внутри структуры осуществляется привычным для высокоуровневых языков способом - через точку. Создадим три переменных для хранения двух структур типа дробь с которыми будем совершать операции, и одну для хранения результата. Инициализируем переменные какими-нибудь значениями. *Опишем целочисленные значения, опишем делимое для обеих дробей и опишем делитель для обеих дробей. Для простоты будем использовать простые дроби типа 1/5.*  В комментарии к каждой дроби я напишу, как бы она выглядела на бумаге.  Внутрь функции структуры данных можно передавать как по значению, так и по ссылке. | int main(int argc, const char\* argv[]){  Fraction f1, f2, result;  f1.integer = -1;  f1.divisible = 1; //-1 | 1 /5  f1.divider = 5;  f2.integer = 1;  f2.divisible = 1; ; //1 | 1 /5  f2.divider = 5;  result.divisible = 0;  result.divider = 0;  } |
| Опишем функцию, которая будет выводить нашу дробь на экран. В эту функцию мы можем передать нашу структуру по значению. Т.е. внутри каждой функции мы будем создавать копию структуры типа дробь, и заполнять её теми значениями, которые передадим в аргументе. Вывод каждой дроби на экран будет зависеть от ряда условий. Именно эти условия мы и опишем внутри нашей функции.  *Если делимое равно 0, то у дроби надо вывести только целую часть, если же делимое не равно 0 и целая часть равно 0 – выводим только дробную часть. Пишем: если делимое не равно 0 то вступает в силу следующее условие – если целая часть равна 0, то печатаем дробь следующим образом: число, значек дроби, число – числа это делимое и делитель.*  *printf("%d / %d", f.divisible, f.divider);*  *В противном случае, если целая часть и делимое не равны 0, то выводим всю дробь – целую часть, делимое и делитель*  *printf("%d %d/%d",f.integer,f.divisible,f.divider);*  *И еще один else для общего if – если делимое равно 0 то выводим только целую чать:*  *printf("%d", f.integer);* | void frPrint(Fraction f) {  if (f.divisible != 0)  if (f.integer == 0)  printf("%d / %d", f.divisible, f.divider);  else  printf("%d %d/%d",f.integer,f.divisible,f.divider);  else  printf("%d", f.integer);  } |
| Проверим, насколько хорошо мы написали нашу функцию? Для этого вызовем ее и передадим туда значения наших дробей.  Добавим пустую строчку и запустим. | frPrint(f1);  puts(“”);  frPrint(f2); |
| Выглядит неплохо, для полноты картины не хватает только научиться выполнять с этими дробями какие-нибудь действия. Для примера возьмём что-то простое, вроде умножения. Передадим в эту функцию значения наших двух дробей и указатель на структуру, в которую будем складывать результат вычислений.  *Назовем нашу функцию frMul, передадим туда необходимые аргументы и немного вспомним математику. Для того чтобы перемножить две дроби нам надо привести их к простому виду, т.е. лишить целой части а затем перемножить числители и знаменатели. Для перевода в простой вид опишем функцию* frDesinteger, в которую будем передавать адрес первой и второй дроби.  Чтобы не перепутать локальные структуры функции и указатели на внешние структуры, доступ к полям внутри указателей на структуры получают не при помощи точки, а при помощи вот такой стрелки. *Т.е. поскольку result для функции frMul является указателем, то мы будем записывать результат не в локальную структуру, а непосредственно в ту структуру, которую мы объявили в в функции Мэйн и передали ссылку на нее нашей функции.*  Ну а далее напишем операции умножения для числителя и знаменателя:  result->divisible = f1.divisible \* f2.divisible;  result->divider = f1.divider \* f2.divider; | void frMul(Fraction f1, Fraction f2, Fraction \*result) {  frDesinteger(&f1);  frDesinteger(&f2);  result->divisible = f1.divisible \* f2.divisible;  result->divider = f1.divider \* f2.divider;  }  void frDesinteger(Fraction \*f) {  f->divisible = f->divisible + (f->integer \* f->divider);  f->integer = 0;  } |
| *Теперь можем выводить результат умножения на экран. Для этого вызовем нашу функцию frMul в которую передадим дробь№1, дробь №2 и адрес на результирующую дробь. Затем вызовем функцию печати frPrint и передадим туда нашу результирующую дробь. Запустим нашу программу и убедимся что все работает корректно.* | puts(“”);  frMul(f1, f2, &result);  frPrint(result); |
| Полученных знаний нам хватит для любых операций со структурами. До встречи на следующем уроке, коллеги. |  |
|  |  |
|  |  |
|  |  |