Лекция 2. Структуры в C++

Казахский национальный университет имени аль-Фараби

Кафедра компьютерных наук

Объектно-ориентированное программирование

Лектор: старший преподаватель, PhD, Кенжебек Ержан Ғалымжанұлы

Цель лекции

Цель лекции

Изучить структуры языка C++ как способ объединения разнородных данных в один составной тип и подготовиться к изучению классов и объектов.

Ожидаемые результаты

1. Определять структуру и создавать переменные структурного типа
2. Обращаться к полям структуры и инициализировать их
3. Выполнять присваивание структурных переменных
4. Объяснить связь структур с классами

План лекции

- Структуры и создание
- Синтаксис и выделение памяти
- Доступ к полям структуры и детали
- Пример применения

Структуры

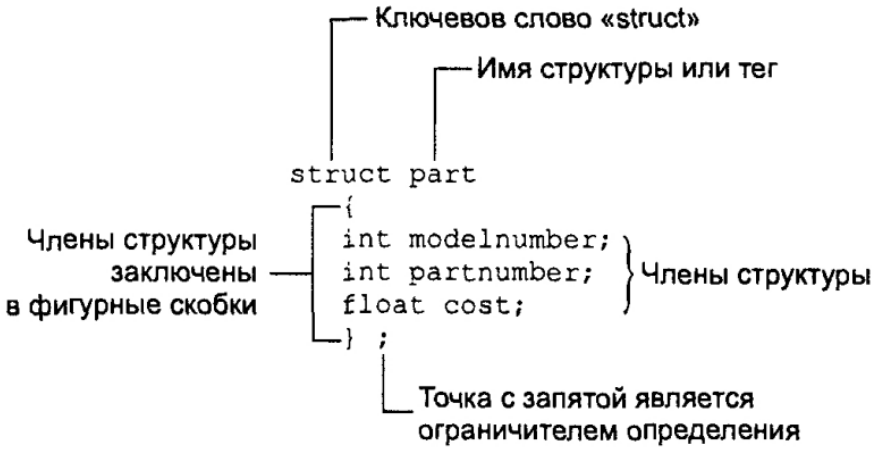
- Структура является объединением простых переменных. Эти переменные могут иметь различные типы: int, float и т. д. Переменные, входящие в состав структуры, называются **полями структуры.**
- Для тех, кто изучает язык C++, структуры являются одной из составляющих главных концепций языка — объектов и классов. На самом деле синтаксис структуры фактически идентичен синтаксису класса. На практике отличие структуры от класса заключается в следующем: структуры, как правило, используют в качестве объединения данных, а классы — в качестве объединения данных и функций. **Таким образом, изучая структуры, мы тем самым закладываем основы для понимания классов и объектов.**

Создание структуры

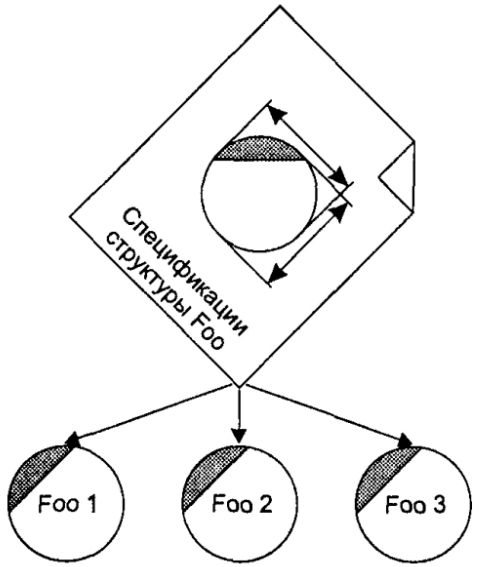
```
// структура для хранения информации о деталях изделий
#include <iostream>
using namespace std;
////////////////////////////////////////////////////////
struct part                  // объявление структуры
{
   int modelnumber;          // номер модели изделия
   int partnumber;           // номер детали
   float cost;               // стоимость детали
};
////////////////////////////////////////////////////////
int main()
{
   part part1;                // определение структурной переменной
   part1.modelnumber = 6244; // инициализация полей
   part1.partnumber = 373;   // переменной part1
   part1.cost = 217.55F;
                             // вывод значений полей на экран
   cout << "Модель " << part1.modelnumber;
   cout << ", деталь " << part1.partnumber;
   cout << ", стоимость $" << part1.cost << endl;
   return 0;
}
```

Вывод: Модель 6244, деталь 373, цена $217.55

Синтаксис и определения структуры



Определение структуры part необходимо для того, чтобы создавать на его основе переменные типа part. Само определение не создает никаких переменных; другими словами, не происходит ни выделения физической памяти, ни объявления переменной. В то же время определение обычной переменной предполагает выделение памяти под эту переменную. Таким образом, определение структуры фактически задает внутреннюю организацию структурных переменных после того, как они будут определены. Следующий рисунок иллюстрирует вышесказанное:



Определение структуры Foo

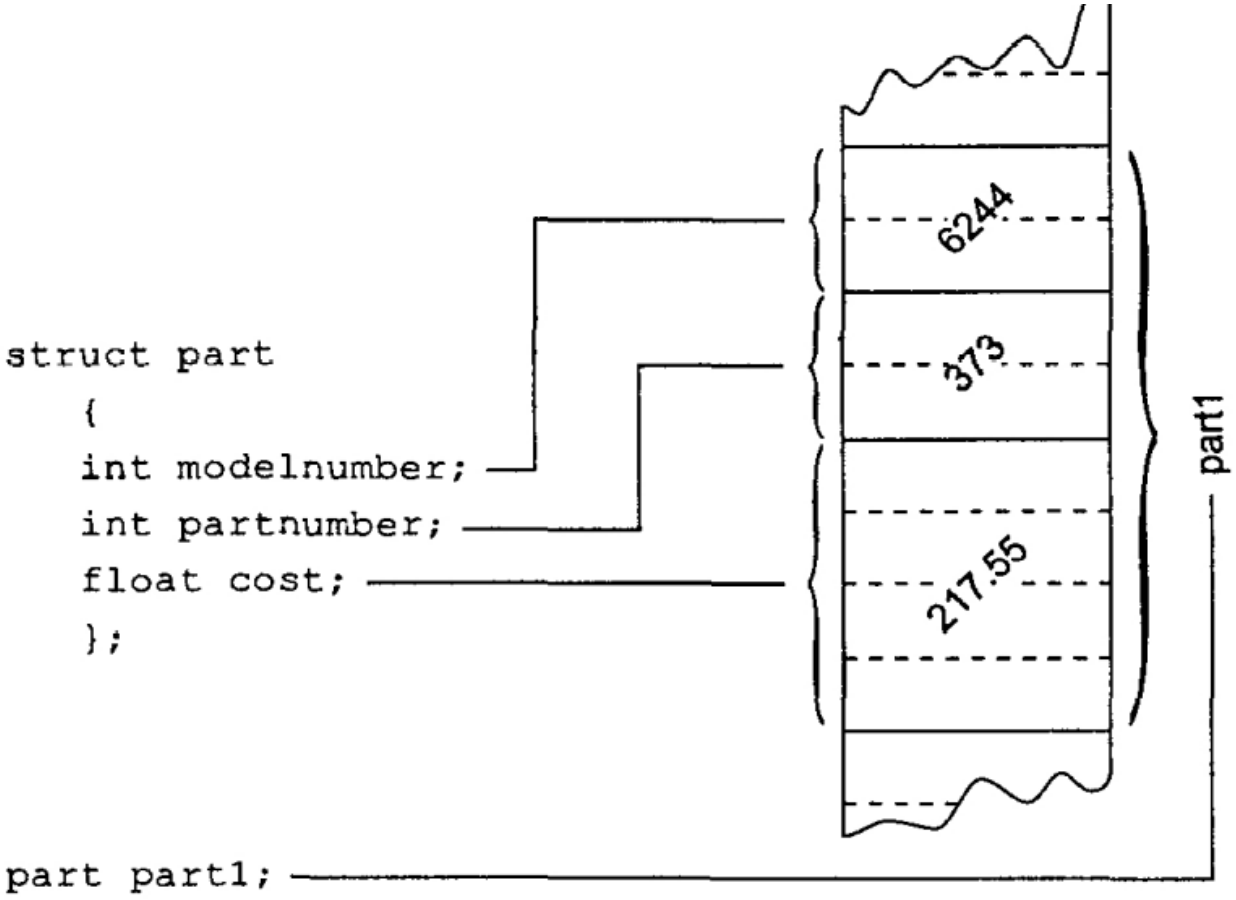
Первый оператор функции main() выглядит следующим образом:

```
part part1;
```

Он представляет собой определение переменной part1, имеющей тип part. Определение переменной означает, что под эту переменную выделяется память.

Сколько же памяти выделяется в данном случае?

Выделение памяти при создание структурированной переменной



В нашем примере необходимо выделить по 8 байта для двух полей типа int и 4 байта для поля типа float

Структура part на самом деле является новым типом данных.

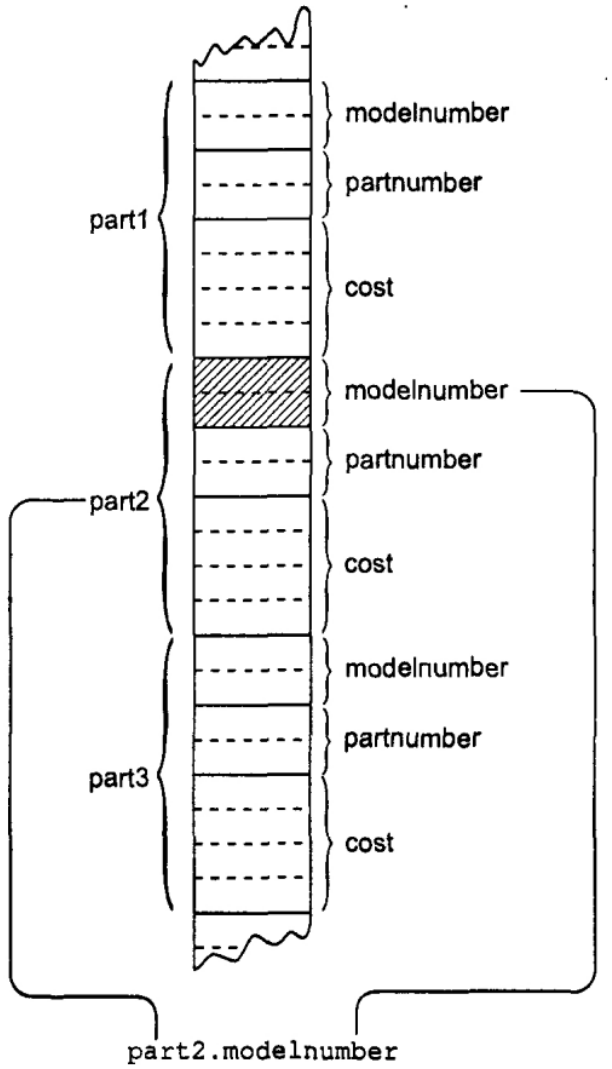
```
part part1;
int var1;
```

Доступ к полям структуры

- Когда структурная переменная определена, доступ к ее полям возможен с применением пока неизвестной нам операции точки. В программе первому из полей структуры присваивается значение при помощи оператор:

```
part1.modelnumber = 6244;
```

- Поле структуры идентифицируется с помощью трех составляющих: имени структурной переменной part1, операции точки (.) и имени поля modelnumber.



Детали

- Можно инициализировать предварительно поля структуры:

```
part part1 = { 6244, 373, 217.55F };
part part2;
```

- Присваивание структуры переменных:

```
part2 = part1;
```

- Обратите внимание на то, что операция присваивания может быть выполнена только над переменными, имеющими один и тот же тип. В случае попытки выполнить операцию присваивания над переменными разных типов компилятор выдаст сообщение об ошибке

Пример применения структур

- Возможно, вы знаете, что в английской системе мер основными единицами измерения длины служат фут и дюйм, причем один фут равен 12 дюймам. Расстояние, равное 15 футам и 8 дюймам, записывается как 15'-8". Дефис в данной записи не означает знак «минус», а служит для разделения значений футов и дюймов

- Предположим, вам необходимо создать инженерный проект с использованием английской системы мер. Было бы удобно представлять длины в виде двух чисел, одно из которых равнялось бы количеству футов, а другое — количеству дюймов.

- Обратите внимание на то, что мы не можем сложить две длины следующим образом:

```
d3 = d1 + d2; // некорректное действие
```

- Такое действие вызовет ошибку потому, что в C++ не определены действия, которые необходимо выполнить при сложении переменных типа Distance. Операция + применима для стандартных типов C++, но она не определена для типов данных, которые создаются пользователями (**пройдем по теме перегрузок операций**)

```
// английская система мер, реализованная с помощью структуры
#include <iostream>
using namespace std;
////////////////////////////////////////////////////////////
struct Distance               // длина в английской системе
{
  int feet;
  float inches;
};
////////////////////////////////////////////////////////////
int main()
{
  Distance d1, d3;            // определения двух переменных
  Distance d2 = { 11, 6.25 }; // определение с инициализацией
                              // ввод полей переменной d1
  cout << "\nВведите число футов: "; cin >> d1.feet;
  cout << "Введите число дюймов: "; cin >> d1.inches;

  // сложение длин d1 и d2 и присвоение результата d3
  d3.inches = d1.inches + d2.inches;  // сложение дюймов
  d3.feet = 0;                        // с возможным заемом
  if(d3.inches >= 12.0)               // если сумма больше 12.0,
  {                                   // то уменьшаем
   d3.inches -= 12.0;                 // число дюймов на 12.0 и
   d3.feet++;                         // увеличиваем число футов на 1
  }

  d3.feet += d1.feet + d2.feet;      // сложение футов
                                     // вывод всех длин на экран
  cout << d1.feet << "\'-" << d1.inches << "\" + ";
  cout << d2.feet << "\'-" << d2.inches << "\" = ";
  cout << d3.feet << "\'-" << d3.inches << "\"\n";
  return 0;
}
```

Выводы

1. Структура объединяет переменные разных типов под одним именем.
2. Определение структуры задаёт новый тип данных; память выделяется при создании переменной этого типа.
3. Доступ к полям осуществляется с помощью операции точки: переменная.поле.
4. Структуры одного типа можно инициализировать списком значений и присваивать друг другу.
5. Для структур не определены арифметические операции (например, d1 + d2); эту проблему решают классы и перегрузка операторов.

Контрольные вопросы

1. Что такое структура? Чем она отличается от массива?
2. Как определить структуру и создать переменную структурного типа?
3. Выделяется ли память при определении структуры? Когда она выделяется?
4. Как получить доступ к полю структуры? Как инициализировать структуру?
5. Можно ли присвоить одну структурную переменную другой? При каком условии?
6. Почему структуры считают предшественниками классов?

Список литературы

Основная литература

1. Лафоре Р. Объектно-ориентированное программирование в C++. 4-е изд. — СПб.: Питер, 2004.
2. Страуструп Б. Язык программирования C++. Специальное издание. — М.: Бином, 2011.
3. Липпман С., Лажойе Ж., Му Б. Язык программирования C++. Базовый курс. 5-е изд. — М.: Вильямс, 2014.

Дополнительная литература

4. Шилдт Г. Полный справочник по C++. 4-е изд. — М.: Вильямс, 2006.
5. Страуструп Б. Программирование: принципы и практика с использованием C++. 2-е изд. — М.: Вильямс, 2016.
6. Stroustrup B. A Tour of C++. 3rd ed. — Addison-Wesley, 2022.

Электронные ресурсы

7. Учебник по C++ (learncpp): www.learncpp.com
8. Справочник по C++ и стандартной библиотеке: en.cppreference.com

- Спасибо за внимание!