

---

# Лекция 2

---

## Структуры и перечисления

---

# Понятие программной структуры

Структура - это композитный тип данных, объединяющий несколько более простых данных. В качестве элементов структуры могут выступать различные типы: `int`, `float`, `bool`, и т.д.

В C++ понятие структуры расширено до нового (пользовательского) типа данных.

Структуры отличаются от массивов тем, что объединяют разнотипные данные.

---

# 1) Синтаксис определения структуры

```
struct имя_структуры
{
    тип_1 поле_1;
    тип_2 поле_2;
    . . . .
    тип_N поле_N;
};
```

Задаются имена и типы элементов (полей) структуры, а также определяется ее имя.

## 2) Синтаксис объявления структурной переменной

```
имя_структуры имя_переменной;
```

При объявлении переменной под нее выделяется участок оперативной памяти. Объем памяти зависит от количества полей в структуре и от их типа.

### 3) Доступ к полям структуры

Для доступа к отдельному полю структуры используется оператор "точка":

```
имя_переменной.поле_k = значение;
```

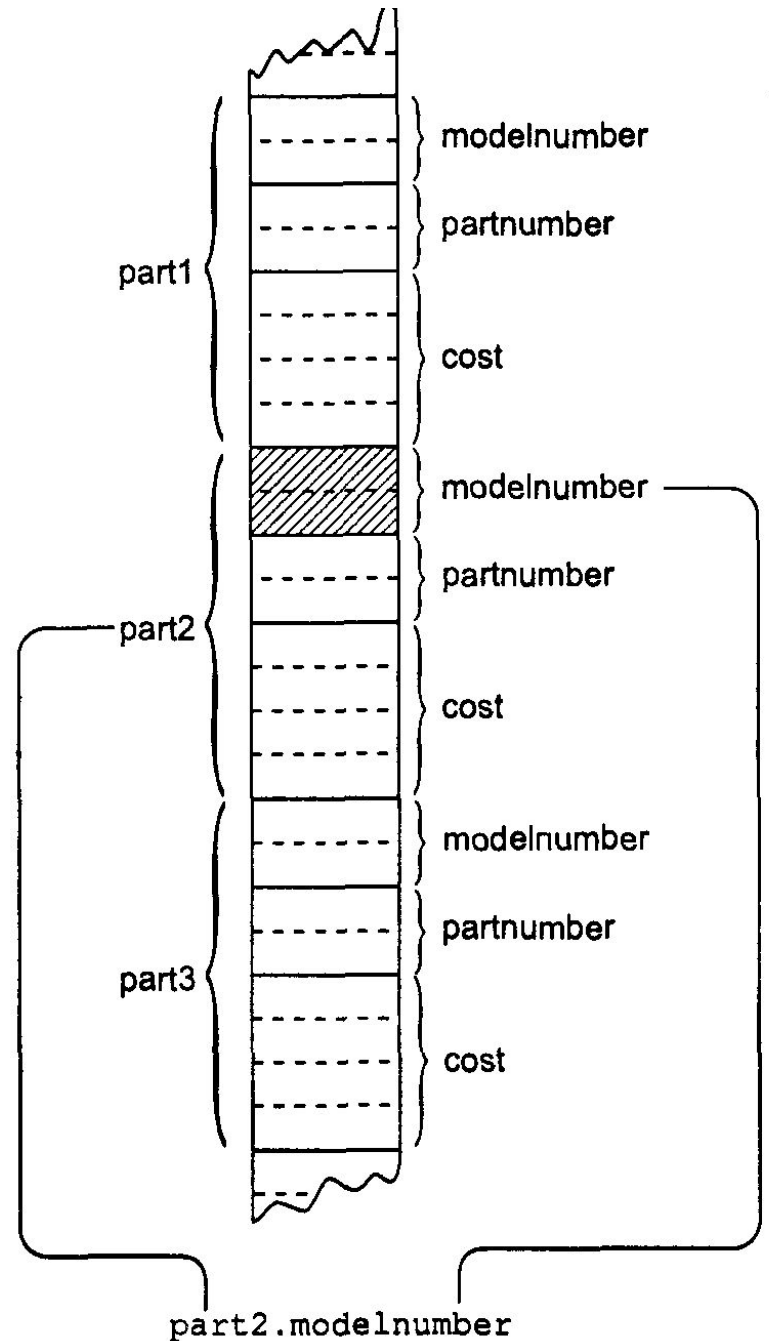
С полями структурной переменной можно обращаться так же, как с обычными переменными (выводить на экран, передавать в функции в качестве аргументов и т.д.).

```
#include <iostream>
using namespace std;

struct part          // объявление структуры
{
    int modelnumber; // номер модели изделия
    int partnumber;  // номер детали
    float cost;      // стоимость детали
};
...
```

# Пример с доступом к полю структурной переменной

```
part part1;  
part part2, part3;  
part2.modelnumber = 28;
```



```
// ...
```

```
int main()
```

```
{
```

```
    // объявление структурной переменной  
    part part1;
```

```
    // присваиваем значения полям  
    part1.modelnumber = 6244;  
    part1.partnumber = 373;  
    part1.cost = 217.55F;
```

```
    // выводим значения полей на экран  
    cout << "Модель " << part1.modelnumber;  
    cout << ", деталь " << part1.partnumber;  
    cout << ", стоимость " << part1.cost << endl;  
    return 0;
```

```
}
```



## 4) Инициализация полей структуры

При объявлении структурной переменной значения ее полей не определены. Для инициализации используется оператор присваивания

```
имя_структуры имя_переменной =  
    { знач_1, знач_2, ..., знач_N };
```

Пример

```
part part1 = {6244, 373, 213.55F};
```

## 5) Присваивание структурных переменных

Операция присваивания, выполняемая над двумя структурными переменными, означает копирование значений всех полей

```
переменная_2 = переменная_1;
```

Присваивание может выполняться только над переменными, имеющими один и тот же тип.

---

# Вложенные структуры

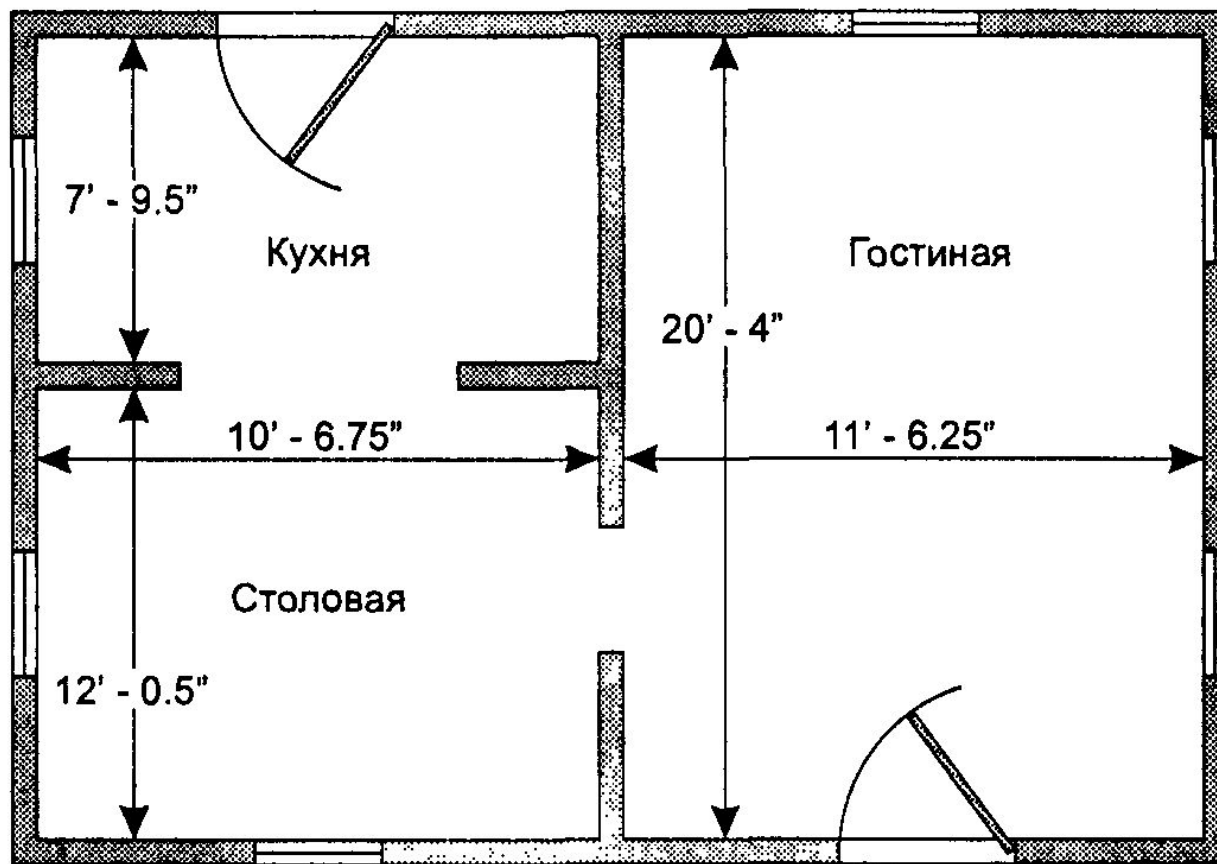
В C++ структуры допускают вложенность.  
Глубина вложения не ограничена.

**Пример:** инженерный проект здания с использованием английской системы мер.

*Исходные данные:*

- 1) Здание = набор помещений
  - 2) Помещение = прямоугольная комната с 2-мя параметрами (длиной и шириной)
  - 3) Единицы длины - фут и дюйм.
-

# Пример: программная структура для описания помещения



Английская система измерения длин – футы и дюймы (1 фут = 12 дюймов)

```
#include <iostream>
using namespace std;

struct Distance          // длина в англ. системе
{
    int feet;
    float inches;
};

struct Room             // комната
{
    Distance length;    // длина
    Distance width;     // ширина
};

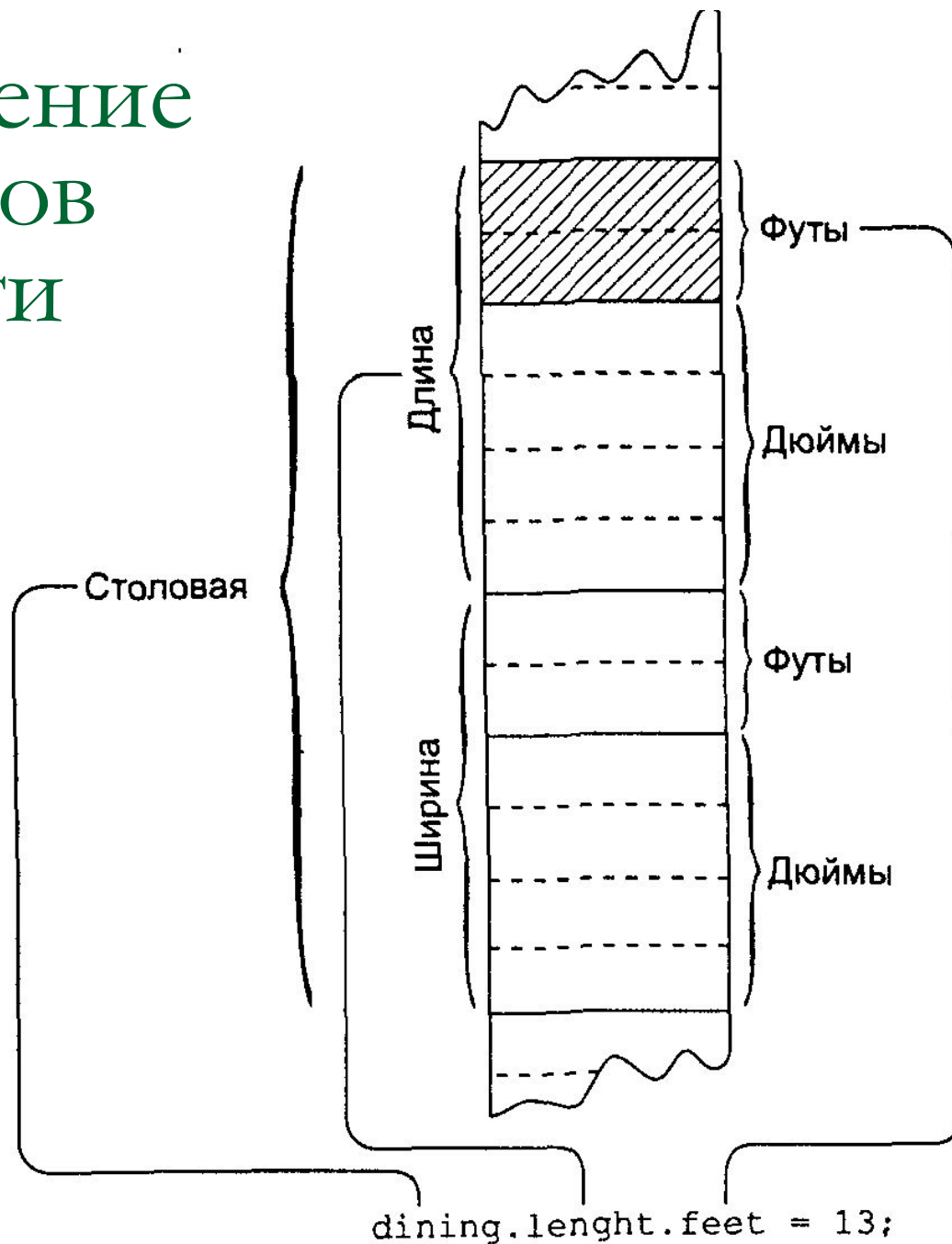
//... (продолжение)
```

```
// ... (начало)

int main()
{
    Room dining;          // комната (столовая)

    dining.length.feet = 13;
    dining.length.inches = 6.5;
    dining.width.feet = 10;
    dining.width.inches = 0.0;
                        // преобразуем
    float l = dining.length.feet +
              dining.length.inches/12;
    float w = dining.width.feet +
              dining.width.inches/12;
                        // площадь комнаты
    cout << "Площадь " << l * w << " кв. футов";
    return 0;
}
```

# Расположение элементов в памяти



---

# Перечисления

Перечисления используются в случаях, когда переменная некоторого типа может принимать заранее известное (как правило, небольшое) множество значений.

**Примеры:** дни недели, оценки на экзамене, шахматные фигуры и т.д.

---



# Синтаксис объявления перечисления

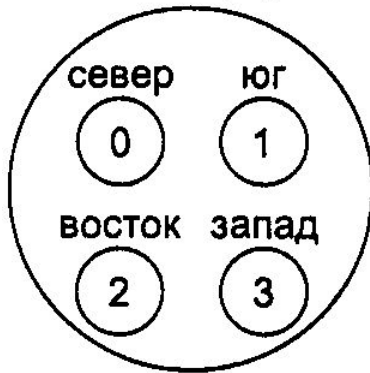
```
enum имя { константа_1, константа_2, ...,  
          константа_N } ;
```

Значение константа\_k называется  
*константой перечислимого типа.*

**Пример:** карточные масти

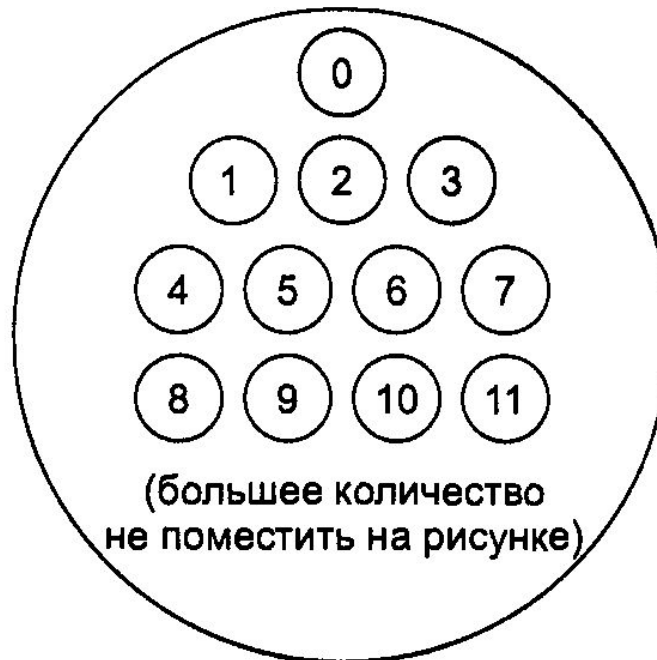
```
enum Suit { Diamonds, Hearts, Clubs, Spades } ;
```

тип enum



Небольшое число значений  
поименовано, и обращение  
к ним происходит по именам

тип int



Когда значений много,  
они не именовются;  
обращение по значениям

# Синтаксис объявления переменной перечисляемого типа

```
имя_перечисления имя_переменной;
```

Примеры объявлений:

```
Suit s1;
```

```
Suit s2, s3 = Diamonds;
```

Переменным перечисляемого типа можно присваивать значения любой из констант, указанных в списке, например:

```
s1 = Hearts;  
s2 = s1;  
s3 = Clubs;
```

Кроме того, перечисляемые типы допускают применение основных арифметических операций и операций сравнения.

```
Suit s1;  
if(s1 == Clubs)  
{  
    cout << "Вы выиграли!";  
}
```

## Приложение: карточная игра.

```
#include <iostream>
using namespace std;

const int jack   = 11;    // валет
const int queen  = 12;    // дама
const int king   = 13;    // король
const int ace    = 14;    // туз

enum Suit {clubs, diamonds, hearts, spades};

struct card      // карта
{
    int number;   // достоинство карты
    Suit suit;    // масть
};

// ... (продолжение)
```

```
// ... (начало)

int main()
{
    card tmp, chosen, prize;        // 3 карты
    int position;

    card card1 = {7, clubs};        // карта 1
    cout << "Карта 1: 7 треф" << endl;

    card card2 = {jack, hearts};    // карта 2
    cout << "Карта 2: валет червей" << endl;

    card card3 = {ace, spades};     // карта 3
    cout << "Карта 3: туз пик\n";

    prize = card3;                 // запоминаем карту 3

    // ... (продолжение)
```

```
// ... (начало)
```

```
cout << "Меняем местами карты 1 и 3\n";  
tmp = card3; card3 = card1; card1 = tmp;
```

```
cout << "Меняем местами карты 2 и 3\n";  
tmp = card3; card3 = card2; card2 = tmp;
```

```
cout << "Меняем местами карты 1 и 2\n";  
tmp = card2; card2 = card1; card1 = tmp;
```

```
cout << "В какой позиции (1, 2, или 3)  
теперь туз пик? ";
```

```
cin >> position;
```

```
// ... (продолжение)
```

```
// ... (начало)
```

```
switch (position)
```

```
{
```

```
    case 1: chosen = card1; break;
```

```
    case 2: chosen = card2; break;
```

```
    case 3: chosen = card3; break;
```

```
}
```

```
if(chosen.number == prize.number &&
```

```
    chosen.suit == prize.suit)
```

```
    cout << "Вы победили!\n";
```

```
else
```

```
    cout << "Вы проиграли.\n";
```

```
return 0;
```

```
}
```