Функции

Лекция 3-3

Определение

- Фу́нкция или подпрограмма — фрагмент программного кода (чаще всего именованный), к которому можно обратиться из другого места программы посредством вызова
- Во время вызова функция выполняет свое тело и возвращает управление в точку вызова

Особенности функций

- Каждая функция должна иметь имя
- Функция должна возвращать значение (если функция возвращает пустое значение void, она может называться процедурой)
- Функция может принимать параметры
- Функция должна быть объявлена и определена
- Функция не выполняется пока не будет явно вызвана
- Функция определяет локальную область видимости, и все переменные, объявленные внутри нее не связаны с глобальными, и будут уничтожены после выхода из функции

Объявление функции

- Объявление функции содержит тип возвращаемого значения, имя и параметры функции. Оно указывает интерфейс функции
- Функция *может* иметь более одного объявления до тех пор пока они идентичны.

```
char input();

void print(int x);

float sum (float x, float y);

int fact(int x);


Неверно – переопределение
возвращаемого значения

float fact(int x);
```

Определение функции

- Определение функции содержит код, который выполняется при вызове функции
- Каждая функция должна иметь только одно определение
- Определение начинается с объявления и заканчивается блоком тела функции
- Компилятор *может* автоматически вставить объявление функции непосредственно перед определением

```
float sum (float x, float y) {
    float z;
    z = x + y;
    return z;
}
```

```
float sum (float x, float y) {
    return x + y;
}
```

Возвращаемое значение

- Возвращаемое значение передается после ключевого слова *return*
- Функция типа *void* должна содержать ключевое слово *return* без значения
- Тип значения после *return* должен быть точно таким, как указано в определении (или будет преобразован к нему, если возможно)
- Компилятор *может* добавить *return* для *void*-функции автоматически
- return не обязательно должен быть последним в функции, особенно если она содержит ветви

```
float sum (float x, float y) {
    ...
    return z;
}

void print (int x) {
    ...
    return;
}
```

Параметры функции

Параметры:

- Обязательные
- Со значением по умолчанию

Параметры передаются в функцию по значению — для каждого вызова функции создается копия всех переменных, которые указаны в ее параметрах

```
void inc (int x) {
    print(x);
    x++;
    print(x);
}

int x = 0;
print(x); // x = 0
inc(x); // x = 0 & x = 1
print(x); // x = 0
```

Область видимости

Область видимости — фрагмент программы, в котором одно и то же имя (переменной или функции) закреплено за одной и той же сущностью.

- Глобальная
- Класса (локальная)
- Функции (локальная)

При обращении к сущности поиск начинается с текущей области видимости и последовательно поднимается вверх.

```
int x = 5;
int y = 7;

void dummy () {
    int x = 3; // маскирует глобальный x
    int z = 9; // локальная
    print(x);
    print(y); // y явно не определен
    print(z);
}

print(x); // -> x = 5
print(y); // -> y = 7
dummy(); // -> x = 3, y = 7, z = 9
print(z); // ошибка: z не определен
```

Параметры со значением по умолчанию

Параметры:

- Обязательные
- Со значением по умолчанию

  Параметры со значениями по умолчанию должны идти после обязательных параметров
- Пользовательские значения по умолчанию должны следовать только по порядку без пропусков

```
float sum (float x, float y = 1) {
    return x + y;
}


sum(1, 2); // -> 3
sum(4); // -> 5
```

Вызов функции

- Вызов функции выполняется указанием ее имени (она должна быть объявлена в коде ранее) и передачей ей обязательных параметров
- Результат вызова функции (*return*) будет подставлен в текущее выражение в место вызова

```
float sum (float x, float y = 1) {
    return x + y;
}


int x = 3;
int y = 5;
sum(sum(x),
    sum(sum(x, 3),
        sum(y, 5))); // 20
```

Рекурсия

- Функция может вызывать сама себя
- При рекурсии обязательно должна быть ветвь, которая не вызывает повторную рекурсию, иначе функция попадет в бесконечную рекурсию и вызовет ошибку
- Частая проблема рекурсии — переполнение стека

```
int fact (int x) {
    if (x > 1) {
        return x * fact(x – 1);
    }
    else {
        return 1;
    }
}
```