Значения аргументов функций по умолчанию

**Аргумент по умолчанию** – это такой аргумент функции, который программист **может не указывать при вызове** функции. Аргумент по умолчанию добавляется компилятором **автоматически.**

Например, ниже объявляется функция f(), имеющая аргументом целую переменную, значением которой по умолчанию является величина 1:

void foo(int i = 1) {

// Тело функции

}

Теперь функция foo() может вызываться двумя способами, как показано ниже:

foo(10); // явная передача значения

foo(); // функция использует значение по умолчанию

В первом случае аргументу i будет присвоено значение 10. Во втором случае i получит значение по умолчанию, равное 1.

**При использовании** аргументов по умолчанию **обратите внимание** на следующие моменты:

* Аргументы по умолчанию используются только в вызовах функции, в которых опущены заключительные **аргументы** — они **должны быть последними** аргументами. Поэтому следующий код недопустим:

int print( double dvalue = 0.0, int prec );

* **Переопределение** аргумента по умолчанию в последующих объявлениях **не допускается**, даже если переопределение совпадает с оригиналом. Поэтому приведенный ниже код вызывает ошибку:

// Prototype for print function.

int print( double dvalue, int prec = 2 );

// ...

// Definition for print function.

int print( double dvalue, int prec = 2 ) {

// ...

}

* **В последующих определениях допускается** добавлять дополнительные аргументы по умолчанию.
* Аргументы по умолчанию могут указываться для указателей на функции.
* Помимо функций аргументы по умолчанию также допускается использовать в конструкторах и методах класса. Пример:

// класс, который описывает круг

class Circle {

// внутренние переменные класса

double x, y, radius;

public:

// конструктор, который принимает аргументы по умолчанию

Circle(double nx=10, double ny = 10, double nradius = 1) {

x = nx;

y = ny;

radius = nradius;

}

// Методы доступа

// чтение значений x, y, r

double GetX(void) { return x; }

double GetY(void) { return y; }

double GetR(void) { return radius; }

// метод, который содержит аргументы по умолчанию

void SetXYR(double nx=0, double ny=0, double nradius=1) {

x = nx;

y = ny;

radius = nradius;

}

};

Во время **объявления** функции, аргументы по умолчанию **задаются** **только один раз**. Здесь возможны два случая. Первый, если функция имеет **прототип и реализацию**. Второй случай, если функция не имеет прототипа, а имеет **только реализацию**.

Пример:

// объявление аргументов по умолчанию в прототипе функции

double GetAreaCircle(double r = 1);

// функция не имеет прототипа, поэтому аргумент по умолчанию задается непосредственно в реализации

double GetVolumeSphere(double r = 1){

return 4.0/3 \* Pi\*r\*r\*r;

}

// реализация функции - аргументы по умолчанию уже заданы в прототипе

double GetAreaCircle(double r) {

return Pi \* r \* r;

}

**Преимущества** использования аргументов по умолчанию:

* **сокращается листинг** программного кода за счет избегания написания лишних функций, которые выполняют ту же работу только с другими значениями аргументов;
* обеспечивается **простой, естественный и эффективный стиль** программирования;
* в некоторых случаях аргументы по умолчанию есть сокращенной формой перегрузки функции. Это, в свою очередь улучшает читабельность программного кода и **упрощает вызов** функции.

Источники:

* <https://www.bestprog.net/ru/2018/07/30/functions-arguments-by-default-in-functions_ru/>
* <https://www.c-cpp.ru/books/znacheniya-argumentov-funkcii-po-umolchaniyu>
* <https://learn.microsoft.com/ru-ru/cpp/cpp/default-arguments?view=msvc-170>

Подготовил **Занин Вячеслав ВМК-21**