# ЛАБОРАТОРНАЯ РАБОТА № 17

# РАЗРАБОТКА И ОТЛАДКА АЛГОРИТМОВ И ПРОГРАММ ПО РЕАЛИЗАЦИИ ПЕРЕГРУЗКИ ФУНКЦИЙ

Цель работы:

* Практическое освоение перегрузки функций.

## КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Часто бывает удобно, чтобы функции, реализующие один и тот же алгоритм для различных типов данных, имели одно и то же имя. Если это имя мнемонично, то есть несет нужную информацию, это делает программу более понятной, поскольку для каждого действия требуется помнить только одно имя. Использование нескольких функций с одним и тем же именем, но с различными типами параметров, называется перегрузкой функций.

Компилятор определяет, какую именно функцию требуется вызвать, по типу фактических параметров. Этот процесс называется разрешением перегрузки (перевод английского слова resolution в смысле «уточнение»). Тип возвращаемого функцией значения в разрешении не участвует. Механизм разрешения основан на достаточно сложном наборе правил, смысл которых сводится к тому, чтобы использовать функцию с наиболее подходящими аргументами и выдать сообщение, если такой не найдется. Допустим, имеется четыре варианта функции, определяющей наибольшее значение:

// Возвращает наибольшее из двух целых:

int max(int, int);

// Возвращает подстроку наибольшей длины:

char\* max(char\*, char\*);

// Возвращает наибольшее из первого параметра и длины второго:

int max (int, char\*);

// Возвращает наибольшее из второго параметра и длины первого: int max (char\*, int);

void f(int a, int b, char\* c, char\* d){

cout << max (a, b) << max(c, d) << max(a, c) << max(c b);}

При вызове функции max компилятор выбирает соответствующий типу фактических параметров вариант функции (в приведенном примере будут последовательно вызваны все четыре варианта функции).

Если точного соответствия не найдено, выполняются продвижения порядковых типов в соответствии с общими правилами, например, bool и char в int, float в double и т. п. Далее выполняются стандартные преобразования типов, например, int в double или указателей в void\*. Следующим шагом является выполнение преобразований типа, заданных пользователем (об этих преобразованиях рассказывается позже), а также поиск соответствий за счет переменного числа аргументов функций. Если соответствие на одном и том же этапе может быть получено более чем одним способом, вызов считается неоднозначным и выдается сообщение об ошибке.

Неоднозначность может появиться при:

* преобразовании типа;
* использовании параметров-ссылок;
* использовании аргументов по умолчанию.

Пример неоднозначности при преобразовании типа:

#include <iostream>

using namespace std;

float f(float i)

{ cout << "function float f(float i)" << endl;

return i; }

double f(double i){

cout << "function double f(double i)" << endl;

return i\*2;

}

int main(){

float x = 10.09;

double у = 10.09;

cout << f(x) << endl; // Вызывается f(float)

cout << f(y)<< endl; // Вызывается f(double)

/\* cout << f(10) << endl; Неоднозначность – как преобразовать 10: во float или double? \*/

return 0;

}

Для устранения этой неоднозначности требуется явное приведение типа для константы 10. Пример неоднозначности при использовании параметров-ссылок: если одна из перегружаемых функций объявлена как int f(int a, int b), а другая — как int f (int a, int &b), то компилятор не сможет узнать, какая из этих функций вызывается, так как нет синтаксических различий между вызовом функции, которая получает параметр по значению, и вызовом функции, которая получает параметр по ссылке. Пример неоднозначности при использовании аргументов по умолчанию:

#include <iostream>

using namespace std;

int f(int a){

return a;}

int f(int a, int b = 1){

return a \* b;}

int main(){

cout << f(10, 2); // Вызывается f(int, int)

/\* cout << f(10); Неоднозначность - что вызывается: f(int, int) или f(int) ? \*/

return 0;

}

Правила описания перегруженных функций.

* Перегруженные функции должны находиться в одной области видимости, иначе произойдет сокрытие аналогично одинаковым именам переменных во вложенных блоках.
* Перегруженные функции могут иметь параметры по умолчанию, при этом значения одного и того же параметра в разных функциях должны совпадать. В различных вариантах перегруженных функций может быть различное количество параметров по умолчанию.
* Функции не могут быть перегружены, если описание их параметров отличается только модификатором const или использованием ссылки (например, int и const int или int и int&).

## ЗАДАНИЕ

Используя ЛР№7(одно задание, по вариантам), сделайте массив разного типа, инициализируйте его через функцию, реализуйте перегрузку функций для своего варианта.

## КОНТРОЛНЫЕ ВОПРОСЫ

Дайте определение понятию перегрузки функции.

В каких ситуациях может возникнуть неоднозначность при перегрузке функций?