Оператор () можно реализовать только как функцию-член. Он может иметь любое число параметров любого типа, тип возвращаемого значения также произвольный. Классы, с перегруженным оператором (), называются функциональными, их экземпляры называются функциональными объектами или функторами.

Пример объявления:

int X::operator() (int a, int b);

Псевдопеременной может быть функция или функтор, вызов которой стоит слева от оператора присваивания и возвращает тип по ссылке.

Пример:

int& foo(int a)

{

return a;

}

Или

class X

{

int a, b;

public:

X& operator() (int x, int y)

{

a = x;

b = y;

return \*this;

}

X& operator= (int x)

{

a = b = x;

return \*this;

}

};

Подстроки могут служить для замены части строки на другую строку, например: есть первая строка содержит «abcdef», вторая — «123456», делаем замену двух символов, начиная с третьего элемента, в первой строке на вторую строку, в итоге получаем в первой строке: «ab123456ef».

Пример кода, предполагаем что есть готовый класс String, в котором перегружена операция() и все данные корректны и размера массива хватает:

class substring

{

char\* beg;

int space;

int& size;

public:

substring(char\* b, int sp, int& sz) : beg(b), space(sp), size(sz) {}

void operator= (String& str)

{

if (str.size < space) //(str.size — размер str без учёта символа конца строки)

{

for (int i = 0; i < str.size; ++i)

beg[i] = str.arr[i]; //(str.arr — указатель на начало строки)

for (int i = str.size, j = space; j < size; ++i, ++j)

beg[i] = beg[j];

size -= space - str.size;

}

else if (str.size > space)

{

for (int i = size + 1 + (str.size - space), j = size + 1; j != space - 1 ; --i, --j)

beg[i] = beg[j];

for (int i = 0; i < str.size; ++i)

beg[i] = str.arr[i];

size += str.size - space;

}

else

{

for (int i = 0; i < space; ++i)

beg[i] = str.arr[i];

}

}

};