

4.1 Параметры компилятора и объяснение их действия. Примеры предупреждений компилятора, не являющихся ошибками с точки зрения языка. Пример неожиданного поведения компилятора при UB с переполнением `int` и включенном параметре `-O`. Объяснение этого поведения.

Параметры компилятора:

- **-Wall** Включает все предупреждения, в том числе предупреждения, отключенные по умолчанию.
- **-Wextra** Некоторые предупреждения не выводятся даже при использовании параметра выше, поэтому используют этот параметр.
- **-Werror** Сообщает компилятору, чтобы все предупреждения были превращены в ошибки, и при их наличии компиляция прерывалась.
- **-O1, -O2, -O3** Различные уровни оптимизации.
- **-O0** Отключение оптимизации.
- **-std=c++11, -std=c++14, -std=c++17, -std=c++2a** Подключение функционала C++11/14/17/20 соответственно.

Примеры предупреждений компилятора, не являющихся ошибками с точки зрения языка:

```
1 // assignment in a conditional statement
2 int x=3, y=4;
3 if(x=y) {}
4
5 // implicit type conversion
6 int n = 10;
7 for(size_t i = 0; i < n; ++i){}
```

Интересное UB с параметром `-O2`

Рассмотрим пример того, как неопределенное поведение в программе может приводить к неожиданным последствиям. Обратимся к коду ниже:

```
1 for(int i = 0; i < 300; i++) {
2     cout << i << " " << i * 12345678 << endl;
3 }
```

Если скомпилировать этот код без параметра оптимизации, то мы получим, просто 300 чисел (при этом на 174 шаге происходит переполнение и выводятся отрицательные числа). Однако, если скомпилировать данный код с оптимизатором `-O2`, то цикл станет бесконечным. Почему? Компилятор считает, что ввод корректен (прогер не дурак), значит `i` не превосходит 173 (так как иначе происходит переполнение), поэтому оптимизатор заменяет условие `i < 300` на `true` и бинго, у нас бесконечный цикл.