**Зачем нужны побитовые операторы?**

В далёком прошлом компьютерной памяти было очень мало и ею сильно дорожили. Это было стимулом максимально разумно использовать каждый доступный бит. Например, в [**логическом типе данных bool**](https://ravesli.com/urok-34-logicheskij-tip-dannyh-boolean/) есть всего лишь два возможных значения (true и false), которые могут быть представлены одним битом, но по факту занимают целый байт памяти! А это, в свою очередь, из-за того, что переменные используют уникальные адреса памяти, а они выделяются только в байтах. Переменная bool занимает 1 бит, а другие 7 тратятся впустую.

Используя побитовые операторы, можно создавать функции, которые позволят уместить 8 значений типа bool в переменной размером 1 байт, что значительно сэкономит потребление памяти. В прошлом такой трюк был очень популярен. Но сегодня, по крайней мере, в прикладном программировании, это не так.

Теперь памяти существенно больше и программисты обнаружили, что лучше писать код так, чтобы было проще и понятнее его поддерживать, нежели усложнять его ради незначительной экономии памяти. Поэтому спрос на использование побитовых операторов несколько спал, за исключением случаев, когда необходима ну уже максимальная оптимизация (например: научные программы, которые используют огромное количество данных; игры, где манипуляции с битами могут быть использованы для дополнительной скорости; встроенные программы, где память по-прежнему ограничена).

**Есть 6 побитовых операторов**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Оператор** | **Символ** | **Пример** | **Операция** |
| Побитовый сдвиг влево | << | x << y | Все биты в x смещаются влево на y бит |
| Побитовый сдвиг вправо | >> | x >> y | Все биты в x смещаются вправо на y бит |
| Побитовое НЕ | ~ | ~x | Все биты в x меняются на противоположные |
| Побитовое И | & | x & y | Каждый бит в x И каждый бит в y |
| Побитовое ИЛИ | | | x | y | Каждый бит в x ИЛИ каждый бит в y |
| Побитовое исключающее ИЛИ (XOR) | ^ | x ^ y | Каждый бит в x XOR каждый бит в y |

В побитовых операциях следует использовать только [**целочисленные типы данных**](https://ravesli.com/urok-31-tselochislennyj-tip-dannyh-integer/) unsigned*,*так как C++ не всегда гарантирует корректную работу побитовых операторов с целочисленными типами signed.

**Правило: При работе с побитовыми операторами используйте целочисленные типы данных unsigned.**