Числовые типы
Конспект

Целочисленные типы

Числа, записанные в коде в виде цифр, по умолчанию имеют тип int, если не превышают значения в таблице:

Тип	Количество байтов	Количество бит	Диапазон значений
int	4	32	от -2 147 483 648 до 2 147 483 647
unsigned int	4	32	от 0 до 4 294 967 295
int8_t	1	8	от -128 до 127
uint8_t	1	8	от 0 до 255
int16_t	2	16	от -32 768 до 32 767
uint16_t	2	16	от 0 до 65 535
int32_t	4	32	от -2 147 483 648 до 2 147 483 647
uint32_t	4	32	от 0 до 4 294 967 295
int64_t	8	64	от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
uint64_t	8	64	от 0 до 18 446 744 073 709 551 615

Типы с суффиксом _t доступны при подключении библиотеки cstdint.

Целочисленные типы:
- Знаковые (signed) — положительные и отрицательные числа и 0
- Безнаковые (unsigned) — положительные числа и 0

Как выбрать тип
- Если вам хватит размера и диапазона значений int — используйте его.
- Если хотите хранить числа порядка триллиона — нужен int64_t.
- Если нужно сэкономить память на числах — берите типы меньшей размерности: int8_t, int16_t.
- Если ваша программ будет запускаться на неизвестной архитектуре — указывайте int32_t.

Типы с суффиксом _t применяют в современном коде. Их размер определён чётко. В старом коде бывают другие названия целочисленных типов. Их размер чётко не определён, но можно привести соответствия на стандартных архитектурах:

- char — int8_t или uint8_t
- signed char — int8_t
- unsigned char — uint8_t
- short int или short — int16_t
- unsigned short int или unsigned short — uint16_t
- long int или long — int32_t
- unsigned long int или unsigned long — uint32_t
- long long int или long long — int64_t
- unsigned long long int или unsigned long long — uint64_t

Чтобы узнать размер типа или выражения в байтах, примените оператор sizeof. Результат вызова sizeof имеет беззнаковый тип size_t. А чтобы вывести минимальное и максимальное значение любого целочисленного типа, подключите <limits>.

Ограниченность памяти и переполнение

Переполнение происходит, когда значение переменной выходит из диапазона значений указанного типа. Код с переполнением может в любой момент дать сбой.

Как избежать переполнения

- Заранее продумывать каждый шаг, в том числе промежуточные вычисления и неявные преобразования типов;
- Выбирать подходящие типы;
- Явно преобразовывать типы оператором static_cast;
- Настроить компилятор так, чтобы предупреждения он считал ошибками.

Операции с целочисленными типами

Чтобы произвести арифметическую или логическую операцию над разными целочисленными типами, компилятор неявно преобразует их к единому типу.

Правила преобразования типов:
- Все типы меньше int компилятор приводит к int.
- Если размер целочисленных типов ≥ int, меньший тип приводится к большему.
- Если размер типов одинаковый, но один из них беззнаковый, знаковый приводится к беззнаковому.

Если вы забыли правила, вызовите связанную с типом ошибку компиляции. В сообщении об ошибке увидите, как в вашем случае происходит преобразование.

Суффикс u (U) показывает, что литерал в коде относится к типу unsigned int:
- литерал 1 — int;
- литерал 1u — unsigned int.

Техника безопасности

Переполнение может произойти при итерации по вектору циклом for., когда компилятор сравнивает знаковый и беззнаковый тип.

Два способа избежать переполнения
1. Использовать только беззнаковые типы;
2. Приводить беззнаковые типы к знаковым оператором static_cast и следить за размером вектора.

Как ещё избежать переполнения
- Проверьте, что код работает в крайних случаях.
- Не вычитайте из беззнаковых типов или будьте внимательны при вычитании.

Два подхода к выбору типа
1. Следовать семантике значений. Если у переменной по смыслу не бывает отрицательных значений — объявлять её беззнаковой.
Минус: придётся помнить все опасности преобразования знаковых и беззнаковых типов.
2. Приводить все беззнаковые типы к знаковым оператором static_cast.
Минус: static_cast заполонит ваш код.

Перечислимые типы

Перечислимый (перечисляемый) — тип данных с конечным числом упорядоченных именованных значений (перечислителей). Объявляется ключевыми словами enum class. В зависимости от позиции в наборе перечислителям присваиваются целочисленные значения. Каждый enum class считается уникальным типом. Поэтому компилятор не будет проводить операции с перечислителями из разных наборов. Значения одного перечислимого типа сравнивают друг с другом операторами ==, !=, < и >. Значения перечислимых типов могут быть элементами множеств или ключами словарей. Порядок между значениями соответствует порядку их определения при объявлении типа.

Оператор switch

Оператор switch — компактный аналог if, а ветка default — компактный аналог else. В отличие от if, switch не умеет проверять произвольные логические выражения. Но может сравнить заданную переменную или результат выражения с конкретными значениями и выполнить действия в зависимости от того, с каким значением произошло совпадение. default-ветка выполнится, если не подошла ни одна case-ветка.

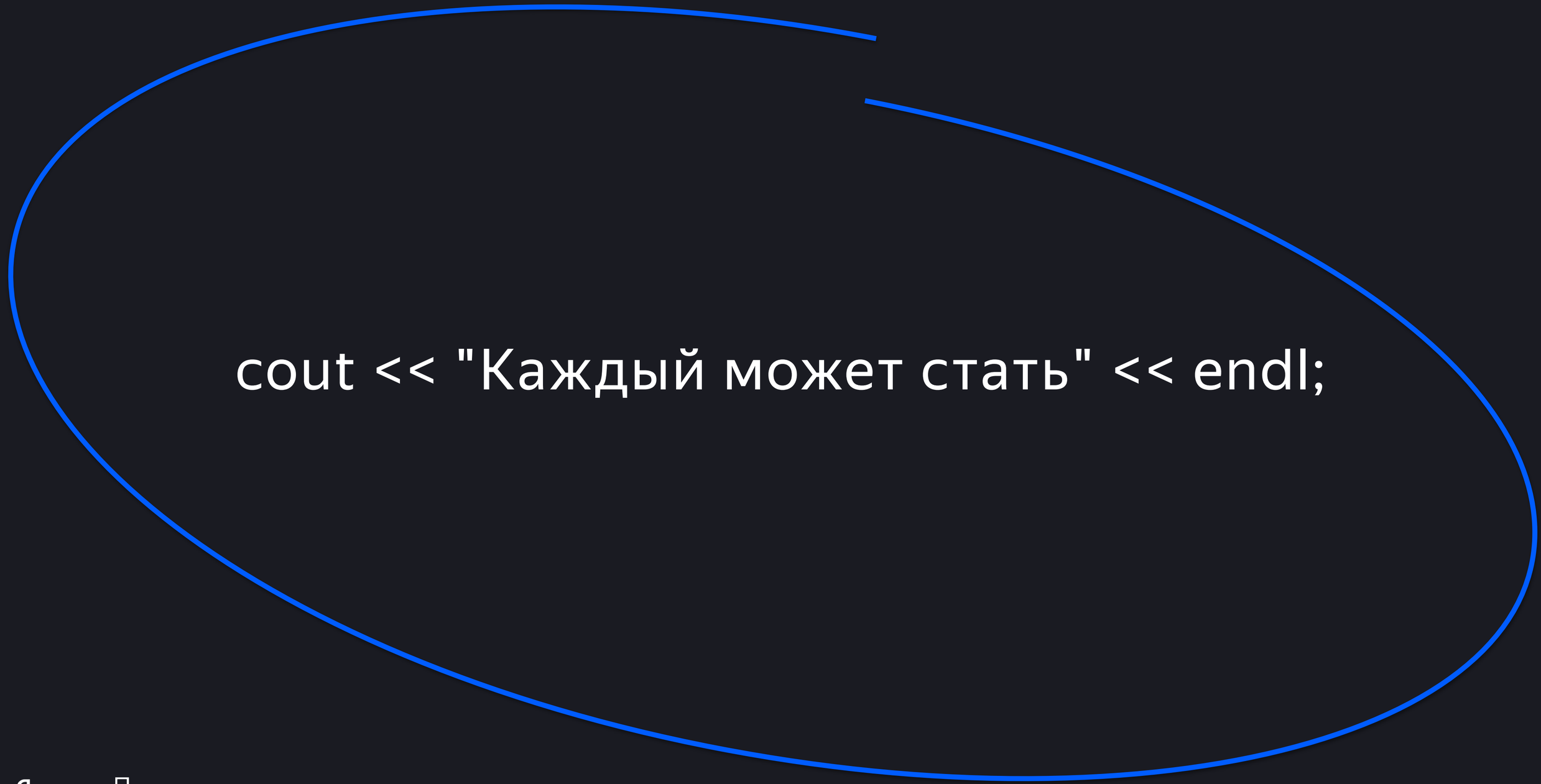
Двойное двоеточие

Двойное двоеточие — оператор разрешения области видимости. У него несколько сфер применения.

Перечисление
Оператор :: позволяет делать значения enum-типа неуникальными в рамках всей программы. Это одно из преимуществ enum class: все имена значений типа «спрятаны» внутрь его имени.

Обращение к сущностям внутри класса
Оператор :: применяют, чтобы снаружи класса обратиться к полю, методу или типу внутри класса.

Пространство имён
Если не написать в начале программы using namespace std, все имена из этого пространства имён нужно употреблять с префиксом std::.


cout << "Каждый может стать" << endl;