



Урок 33. C++: числовая задача с циклом

Часть 1. Материалы ученика

Программирование на Python и C++ · 7 класс

Урок 33 из 34 · 45 минут · Модуль «Марафон: самостоятельный алгоритм»

Уровень: начинающий программист; C++17.

Программа: <https://schoolotzyv.ru/blog/python-cpp-7-klass-programma>

Нужно разложить 58 карточек по пакетам не более 12. Последний пакет неполный. Как программа убедится, что не потеряла и не создала карточки?

Результат урока: Разработать цикл упаковки, проверить сохранение количества и завершение, затем перенести способ на расход разделителей.

01. Проблемная ситуация

За 5 минут выпишите содержимое пакетов для 58 карточек и вместимости 12. Ученик отвечает 4 пакета по 12: найдите потерю. Другой создаёт 5 пакетов по 12: найдите лишние карточки. Сохраните правильную последовательность для теста.

02. Понятия и разобранный пример

5–14 минут: 4 минуты на понятия, 3 на пример, 2 на объяснение способа партнёру.

Состояние, инвариант и строгое уменьшение остатка

В первой задаче n — исходное число карточек; $remaining$ — ещё не упакованные; $count$ — готовые пакеты; $last$ — размер последнего. За шаг берём $capacity$ или оставшийся меньший остаток. До и после каждого шага число уже упакованных плюс $remaining$ равно n .

При $remaining > 0$ и $capacity \geq 1$ $take$ положителен и не превосходит $remaining$, поэтому остаток строго уменьшается и не становится отрицательным. Максимум миллион шагов при $capacity=1$. Выход $remaining=0$ означает, что все карточки распределены; для $n=0$ пакетов нет, $last$ по контракту равен 0.

Независимая проверка: $count=n/capacity$ плюс 1, если остаток не ноль. Если $n>0$, $last=(n-1)\%capacity+1$. Это формулы для проверки цикла, не скрытая замена требуемой реализации. В решении `long long` с ограниченными входами; проверка $capacity$ выполняется до деления.

Отступы набирайте обычными пробелами: 4 на уровень. При копировании с веб-страницы замените неразрывные пробелы обычными; полный код с обычными пробелами есть в PDF.

58→46→34→22→10→0. Пять шагов с $take=12,12,12,12,10$; вывод 5 10. Сумма $4 \cdot 12 + 10 = 58$. Для 24 12 ответ 2 12, а не 3 0; для 0 12 — 0 0.

```
#include <iostream>
```

```
int main() {
    long long n, capacity;
    if (!(std::cin >> n >> capacity) || n < 0 || n > 1000000
        || capacity < 1 || capacity > 1000000) {
        std::cout << "error\n"; return 0;
    }
    long long remaining = n, count = 0, last = 0;
    while (remaining > 0) {
        long long take = capacity;
        if (remaining < take) take = remaining;
        remaining -= take;
        ++count;
        last = take;
    }
    std::cout << count << " " << last << "\n";
}
```



C++: числовая задача с циклом

Старт	remaining=n; count=0; last=0
Выбор take	не больше capacity и remaining
Обновление	remaining-=take; count++; last=take
Проверка	упаковано + remaining = n

58 карточек при вместимости 12: пакеты 12,12,12,12,10; остаток убывает 58,46,34,22,10,0.

03. Практика 1. Проверяем материал

Полная упаковка и независимая арифметическая проверка

14–25 минут. Закройте код, составьте свой алгоритм упаковки с while и независимые ожидаемые ответы.

n capacity	Вывод count last
0 12	0 0
1 12	1 1
24 12	2 12
25 12	3 1
58 12	5 10
5 0	error

2 минуты — состояния; 4 — самостоятельный исходник; 3 — таблица и 1000000 1; 2 — доказательство сохранения и остановки. Продукт: код, семь тестов и две фразы об инварианте. Критерии: карточки сохранены, остаток убывает, точное деление и ноль обработаны, последний пакет указан верно.

04. Практика 2. Перенос способа

Разделитель расходует ресурс, но не является содержимым

25–33 минуты. Новый ресурс stock: каждый пакет требует сначала одну карточку-разделитель и от 1 до capacity карточек содержимого. Разделитель не входит в содержимое. Пока возможно, заполняют пакет максимально; пустые пакеты не делают. Ввод stock=0...1000000, capacity=1...1000000. Вывод count,items,неиспользованный stock. За 2 минуты прогноз 14 5; за 4 код; за 2 тесты 0 5,1 5,2 5,12 5. Критерии: отдельный расход разделителя, нет пустого пакета, общий ресурс сохранён.

05. Обсуждение и границы вывода

33–40 минут. Можно ли перенести формулу первой задачи во вторую без изменений? Защитите независимую проверку инвариантом; найдите контрпример и объясните, какой ресурс был забыт.

06. Индивидуальная проверка

40–45 минут. Для первой задачи 37 8 найдите count,last. Для второй 9 3 выпишите содержимое пакетов, count,items,остаток и проверьте баланс. Четыре критерия: первый ответ, состав, итог, баланс.



Часть 2. Ответы и материалы учителя

Практика 1: разбор и критерии

Для 1000000 1 ответ 1000000 1. Остальные ответы в таблице. Вычитание capacity без ограничения сделало бы отрицательный остаток; while remaining>=0 создал бы лишний пустой пакет либо зациклился при take=0. Полный эталонный код ниже.

```
#include <iostream>

int main() {
    long long n, capacity;
    if (!(std::cin >> n >> capacity) || n < 0 || n > 1000000
        || capacity < 1 || capacity > 1000000) {
        std::cout << "error\n"; return 0;
    }
    long long remaining = n, count = 0, last = 0;
    while (remaining > 0) {
        long long take = capacity;
        if (remaining < take) take = remaining;
        remaining -= take;
        ++count;
        last = take;
    }
    std::cout << count << " " << last << "\n";
}
```

Практика 2: разбор и критерии

14 5: пакеты содержимого 5,5,1; count=3,items=11,остаток 0. 0 5→0 0 0; 1 5→0 0 1; 2 5→1 1 0; 12 5→2 10 0. Инвариант: исходный stock = items+count+текущий stock. Каждый шаг расходует минимум 2, остаток на выходе 0 или 1.

```
#include <iostream>

int main() {
    long long stock, capacity;
    if (!(std::cin >> stock >> capacity) || stock < 0 || stock > 1000000
        || capacity < 1 || capacity > 1000000) {
        std::cout << "error\n"; return 0;
    }
    long long count = 0, items = 0;
    while (stock > 1) {
        long long take = stock - 1;
        if (take > capacity) take = capacity;
        stock -= take + 1;
        items += take;
        ++count;
    }
    std::cout << count << " " << items << " " << stock << "\n";
}
```

Обсуждение: возможный вывод

Для 14 5 простая упаковка дала бы 3 пакета с 14 карточками внутри, но разделители требуют ещё 3. Во второй задаче только 11 внутри и 3 разделителя. Полный журнал одного теста не заменяет границы 0,1 и точное заполнение. Программа должна выполнять контракт, а не лишь совпадать с одним примером.

Индивидуальная проверка: ответ и критерии

37 8→5 5. Во второй пакеты 3 и 3, затем остаётся 1, нового пакета нет: count2,items6,stock1. Баланс 9=6+2+1. По одному баллу за четыре проверяемых элемента.



Учителю: подготовка и план на 45 минут

Подготовьте компьютер, рабочую папку с правом записи, редактор, распечатку ученической части PDF, бумагу и ручку. Локальный компилятор G++ с C++17 или настроенная учителем IDE; сборка `g++ -std=c++17 -Wall -Wextra -pedantic main.cpp -o lesson`. В Windows задайте `lesson.exe`. Запуск `./lesson` в Linux/macOS, `.\lesson.exe` в Windows. Для урока 31 дополнительно обычный браузер: открывать локальный SVG, не C++-исходник. Все входы и ожидаемые результаты даны. Установку и проверку запуска учитель выполняет до занятия; детские аккаунты и интернет не нужны.

Уровень — начинающий программист, 7 класс. Предпосылки: `while`, `if`, счётчик, остаток и проверка границ. В паре автор вводит и объясняет, проверяющий сопоставляет с прогнозом, затем роли меняются. Без компьютера трассируйте полный код на бумаге, отмечайте координаты по схеме и проверяйте готовые результаты; учитель демонстрирует заранее собранный пример. Бумажная трассировка не выдаётся за реальную компиляцию или запуск ученика. План 45 минут включает чтение, ввод, сборку и обсуждение. В графических задачах сначала прогноз, затем сравнение. Матрица программы предназначена учителю.

Минуты	Работа	Результат
0–5	Проблема	Вопрос и первое решение
5–14	Понятия и пример	Способ проверки
14–25	Практика 1	Готовый продукт и проверка
25–33	Практика 2	Перенос способа
33–40	Обсуждение	Аргумент и ограничение
40–45	Индивидуальная проверка	Самостоятельный ответ

07. Соответствие исходной программе

Пункт программы	Действие ученика	Проверяемый результат
Повторение и обобщение (Марафон по программированию)	Разработать цикл упаковки, проверить сохранение количества и завершение, затем перенести способ на расход разделителей.	Самостоятельный продукт и тесты, описанные в практиках
Составлять и отлаживать простые диалоговые программы	Сверяет прогноз с фактическим запуском; объясняет ошибочный вариант	Журнал тестов и индивидуальный ответ по критериям

Источники и права на изображения

Исходная программа курса

<https://schoolotzyv.ru/blog/python-cpp-7-klass-programma>

Рабочий текст стандарта C++: циклы

<https://eel.is/c++draft/stmt.iter>

Тексты, учебные данные, реплики, программы и схемы подготовлены для уроков портала «Школы России». Числовые серии — учебные модели, не результаты проведённых классом опытов. Схемы авторские, сторонние фотографии не используются. Методические ссылки приведены для учителя; ученик получает все материалы здесь. Проверка первичных источников: 17.10.2026.