// C++ · Лабораторная 08

Функции

Два задания: генератор функций с замыканием и простая рекурсивная функция.

Цель и порядок работы

Научиться возвращать лямбду из функции, сохранять настройки в замыкании и писать рекурсию с базовым случаем.

В первом задании обычная функция создаёт и возвращает другую функцию — лямбду с захваченными значениями. Во втором решите маленькую числовую задачу рекурсией: сумма, факториал, степень или работа с цифрами. В каждом шаге достаточно одного рекурсивного вызова.

1. Выполните два задания своего варианта; ввод и вывод оставьте в main.
2. Генератор должен возвращать вызываемую лямбду, а не сразу вычисленное число. Захватывайте настройки по значению.
3. Создайте два замыкания с разными настройками и проверьте каждое на нескольких аргументах.
4. В рекурсивной функции выделите базовый случай и шаг, который приближает к нему.
5. Соблюдайте небольшие диапазоны из условия. Покажите результаты и цепочку рекурсивных вызовов для маленького примера.

Требования к коду, заготовка отчёта, тестирование и блок-схемы →

Перед началом: Лекция 3 · замыкания и генераторы функций · Лекция 3 · области видимости · Лекция 9 · конечная рекурсия и стек · Лекция 2 · циклы

> Возвращайте лямбду через auto и захватывайте настройки явно по значению. Захват ссылки на локальную переменную генератора не продлевает её время жизни.

Задания по вариантам

Выполните все задания назначенного варианта.

Общие требования к вариантам

В каждом варианте два задания. Первое — генератор функций: обычная функция создаёт и возвращает лямбду с захваченными настройками. Второе — короткая рекурсивная функция с одним рекурсивным вызовом на каждом шаге. Сначала напишите рекурсивную функцию, если генератор из варианта+ использует её.

Настройки захватывайте явно по значению: [shift], [a, b] и т. п. Для возвращаемого типа генератора используйте auto (C++14 и новее): компилятор выводит тип лямбды. Не возвращайте захваты по ссылке на локальные переменные генератора. Классы, std::function и указатели на функции в этой работе не требуются.

В main создайте два замыкания с разными настройками. Каждое вызовите минимум для трёх аргументов, включая ноль и границу, если они допустимы. Измените переменную, из которой было взято одно из настроечных значений, и покажите, что уже созданная лямбда сохранила прежнюю копию. Затем ещё раз вызовите первое замыкание, чтобы убедиться в независимости двух объектов.

Для настроек и обычных целых аргументов используйте диапазон от −100 до 100, если вариант не задаёт более строгий. Ограничения положительности, порядка границ и неотрицательного количества проверяйте до вычислений. Ввод и вывод оставьте в main; генератор и рекурсия возвращают результат. Константы объявляйте вне main.

Ниже показан только механизм создания замыкания: makeConstant возвращает функцию без аргументов, запомнившую своё число. Программа выводит 42. В своём варианте верните функцию с указанными параметрами и вычислением.

FRAGMENT.CPP

```cpp
#include <iostream>

constexpr int SAVED_VALUE = 42;

auto makeConstant(int value) {
    return [value]() { return value; };
}

int main() {
    auto getValue = makeConstant(SAVED_VALUE);
    std::cout << getValue() << '\n';
}
```

Для рекурсии сначала запишите базовый случай, затем шаг к нему. Циклы, готовые функции вычисления суммы/факториала/степени и преобразование числа в строку во втором задании не используйте. Проверьте базовый случай, один шаг и несколько шагов; составьте цепочку вызовов для маленького входа. Глубина здесь ограничена входными данными, самое большое n для суммы — 100.

Вариант 1 · Прибавление числа

Задание 1 · Генератор функций

Создайте генератор makeAdder(shift), возвращающий лямбду с одним аргументом x. Лямбда прибавляет сохранённый shift к x.

$$f(x) = x + \text{shift}$$

Пример: makeAdder(5), вызов с 7 → 12. Функция makeAdder(shift) должна вернуть лямбду, а не сразу вычислить один ответ.

Задание 2 · Рекурсия: сумма от 1 до n

Для целого n от 0 до 100 рекурсивно найдите сумму чисел от 1 до n. Для n = 0 результат 0. На каждом шаге уменьшайте n на 1.

$$S(0) = 0, \qquad S(n) = n + S(n-1) \quad (n > 0)$$

Проверка: n = 4 → 10; n = 0 → 0.

Вариант 2 · Умножение на коэффициент

Задание 1 · Генератор функций

Генератор makeMultiplier(factor) возвращает лямбду, умножающую каждый переданный x на сохранённый коэффициент.

$$f(x) = \text{factor} \cdot x$$

Пример: makeMultiplier(3), вызов с 4 → 12. Функция makeMultiplier(factor) должна вернуть лямбду, а не сразу вычислить один ответ.

Задание 2 · Рекурсия: факториал

Для целого n от 0 до 12 рекурсивно вычислите n!. Для n = 0 верните 1. На каждом шаге умножайте n на результат для n − 1. Результат храните в long long.

$$F(0) = 1, \qquad F(n) = nF(n-1) \quad (n > 0)$$

Проверка: n = 5 → 120; n = 0 → 1.

Вариант 3 · Линейная функция

Задание 1 · Генератор функций

Генератор makeLinear(a, b) возвращает лямбду с двумя захваченными коэффициентами. Параметр x передаётся при её вызове.

$$f(x) = ax + b$$

Пример: makeLinear(2, 3), вызов с 4 → 11. Функция makeLinear(a, b) должна вернуть лямбду, а не сразу вычислить один ответ.

Задание 2 · Рекурсия: степень числа

Для целого основания a от −10 до 10 и целого показателя n от 0 до 8 рекурсивно вычислите a в степени n. При n = 0 возвращайте 1, в том числе для a = 0. На каждом шаге уменьшайте показатель на 1. Используйте long long.

$$P(a, 0) = 1, \qquad P(a, n) = aP(a, n-1) \quad (n > 0)$$

Проверка: a = 2, n = 5 → 32; a = −2, n = 3 → −8.

Вариант 4 · Нижняя граница

Задание 1 · Генератор функций

Верните из makeLowerBound(minimum) лямбду: если x меньше сохранённой границы, она возвращает границу, иначе — x.

$$f(x) = \max(x, \text{minimum})$$

Пример: makeLowerBound(5): для 2 → 5, для 8 → 8. Функция makeLowerBound(minimum) должна вернуть лямбду, а не сразу вычислить один ответ.

Задание 2 · Рекурсия: количество цифр

Для неотрицательного целого n не больше 10^9 рекурсивно найдите количество десятичных цифр. Если n меньше 10, ответ равен 1. Иначе уберите последнюю цифру целочисленным делением на 10.

$$D(n) = \begin{cases} 1, & 0 \le n < 10, \\ 1 + D(\lfloor n/10 \rfloor), & n \ge 10. \end{cases}$$

Проверка: n = 0 → 1; n = 507 → 3.

Вариант 5 · Верхняя граница

Задание 1 · Генератор функций

Верните лямбду, ограничивающую результат сверху: x выше сохранённой границы заменяется границей, остальные значения сохраняются.

$$f(x) = \min(x, \text{maximum})$$

Пример: makeUpperBound(5): для 2 → 2, для 8 → 5. Функция makeUpperBound(maximum) должна вернуть лямбду, а не сразу вычислить один ответ.

Задание 2 · Рекурсия: сумма цифр

Для неотрицательного целого n не больше 10^9 рекурсивно найдите сумму десятичных цифр. База: n = 0 даёт 0. На шаге прибавьте последнюю цифру к результату для оставшейся части числа.

$$T(0) = 0, \qquad T(n) = (n \bmod 10) + T(\lfloor n/10 \rfloor) \quad (n > 0)$$

Проверка: n = 507 → 12; n = 0 → 0.

Вариант 6 · Проверка порога

Задание 1 · Генератор функций

Верните лямбду-предикат с результатом bool: x должен быть не меньше захваченного threshold. Покажите случай равенства.

$$f(x) = (x \geq \text{threshold})$$

Пример: makeThreshold(10): для 9 → false, для 10 → true. Функция makeThreshold(threshold) должна вернуть лямбду, а не сразу вычислить один ответ.

Задание 2 · Рекурсия: сумма от 1 до n

Для целого n от 0 до 100 рекурсивно найдите сумму чисел от 1 до n. Для n = 0 результат 0. На каждом шаге уменьшайте n на 1.

$$S(0) = 0, \qquad S(n) = n + S(n-1) \quad (n > 0)$$

Проверка: n = 4 → 10; n = 0 → 0.

Вариант 7 · Расстояние до точки

Задание 1 · Генератор функций

Верните лямбду, вычисляющую расстояние от x до сохранённой точки origin на числовой прямой.

$$f(x) = |x - \text{origin}|$$

Пример: makeDistance(5): для 2 → 3, для 5 → 0. Функция makeDistance(origin) должна вернуть лямбду, а не сразу вычислить один ответ.

Задание 2 · Рекурсия: факториал

Для целого n от 0 до 12 рекурсивно вычислите n!. Для n = 0 верните 1. На каждом шаге умножайте n на результат для n − 1. Результат храните в long long.

$$F(0) = 1, \qquad F(n) = nF(n-1) \quad (n > 0)$$

Проверка: n = 5 → 120; n = 0 → 1.

Вариант 8 · Проверка делимости

Задание 1 · Генератор функций

Для положительного divisor создайте предикат, проверяющий делимость переданного целого x без остатка. Нулевой делитель отклоните до создания лямбды.

$$f(x) = (x \bmod \text{divisor} = 0)$$

Пример: makeDivisibilityTest(3): для 9 → true, для 10 → false. Функция makeDivisibilityTest(divisor) должна вернуть лямбду, а не сразу вычислить один ответ.

Задание 2 · Рекурсия: степень числа

Для целого основания a от −10 до 10 и целого показателя n от 0 до 8 рекурсивно вычислите a в степени n. При n = 0 возвращайте 1, в том числе для a = 0. На каждом шаге уменьшайте показатель на 1. Используйте long long.

$$P(a, 0) = 1, \qquad P(a, n) = aP(a, n-1) \quad (n > 0)$$

Проверка: a = 2, n = 5 → 32; a = −2, n = 3 → −8.

Вариант 9 · Проверка интервала

Задание 1 · Генератор функций

Захватите две упорядоченные границы. Возвращаемый предикат проверяет принадлежность x закрытому интервалу; равные границы разрешены.

$$f(x) = (\text{left} \le x \le \text{right})$$

Пример: makeIntervalCheck(2, 5): для 2 → true, для 6 → false. Функция makeIntervalCheck(left, right) должна вернуть лямбду, а не сразу вычислить один ответ.

Задание 2 · Рекурсия: количество цифр

Для неотрицательного целого n не больше 10^9 рекурсивно найдите количество десятичных цифр. Если n меньше 10, ответ равен 1. Иначе уберите последнюю цифру целочисленным делением на 10.

$$D(n) = \begin{cases} 1, & 0 \le n < 10, \\ 1 + D(\lfloor n/10 \rfloor), & n \ge 10. \end{cases}$$

Проверка: n = 0 → 1; n = 507 → 3.

Вариант 10 · Модуль со смещением

Задание 1 · Генератор функций

Верните лямбду, которая сначала находит модуль переданного x, затем прибавляет сохранённый offset.

$$f(x) = |x| + \text{offset}$$

Пример: makeAbsoluteOffset(3), вызов с −4 → 7. Функция makeAbsoluteOffset(offset) должна вернуть лямбду, а не сразу вычислить один ответ.

Задание 2 · Рекурсия: сумма цифр

Для неотрицательного целого n не больше 10^9 рекурсивно найдите сумму десятичных цифр. База: n = 0 даёт 0. На шаге прибавьте последнюю цифру к результату для оставшейся части числа.

$$T(0) = 0, \qquad T(n) = (n \bmod 10) + T(\lfloor n/10 \rfloor) \quad (n > 0)$$

Проверка: n = 507 → 12; n = 0 → 0.

Вариант 11 · Стоимость покупки

Задание 1 · Генератор функций

Для положительной целой цены верните функцию стоимости. Количество товаров quantity вводится отдельно при вызове лямбды и неотрицательно.

$$f(q) = \text{unitPrice} \cdot q$$

Пример: makePrice(7): для количества 3 → 21, для 0 → 0. Функция makePrice(unitPrice) должна вернуть лямбду, а не сразу вычислить один ответ.

Задание 2 · Рекурсия: сумма от 1 до n

Для целого n от 0 до 100 рекурсивно найдите сумму чисел от 1 до n. Для n = 0 результат 0. На каждом шаге уменьшайте n на 1.

$$S(0) = 0, \qquad S(n) = n + S(n-1) \quad (n > 0)$$

Проверка: n = 4 → 10; n = 0 → 0.

Вариант 12 · Квадрат со смещением

Задание 1 · Генератор функций

Верните лямбду, вычисляющую квадрат переданного x и прибавляющую сохранённый offset.

$$f(x) = x^2 + \text{offset}$$

Пример: makeSquareOffset(2), вызов с −3 → 11. Функция makeSquareOffset(offset) должна вернуть лямбду, а не сразу вычислить один ответ.

Задание 2 · Рекурсия: факториал

Для целого n от 0 до 12 рекурсивно вычислите n!. Для n = 0 верните 1. На каждом шаге умножайте n на результат для n − 1. Результат храните в long long.

$$F(0) = 1, \qquad F(n) = nF(n-1) \quad (n > 0)$$

Проверка: n = 5 → 120; n = 0 → 1.

Вариант 13 · Проверка совпадения

Задание 1 · Генератор функций

Захватите expected по значению. Лямбда возвращает bool: совпадает ли её аргумент с сохранённым значением.

$$f(x) = (x = \text{expected})$$

Пример: makeEqualityCheck(8): для 8 → true, для 7 → false. Функция makeEqualityCheck(expected) должна вернуть лямбду, а не сразу вычислить один ответ.

Задание 2 · Рекурсия: степень числа

Для целого основания a от −10 до 10 и целого показателя n от 0 до 8 рекурсивно вычислите a в степени n. При n = 0 возвращайте 1, в том числе для a = 0. На каждом шаге уменьшайте показатель на 1. Используйте long long.

$$P(a,0)=1, \qquad P(a,n)=aP(a,n-1) \quad (n>0)$$

Проверка: a = 2, n = 5 → 32; a = −2, n = 3 → −8.

Вариант 14 · Изменение знака и сдвиг

Задание 1 · Генератор функций

Создайте лямбду, меняющую знак переданного аргумента и прибавляющую сохранённый offset.

$$f(x)=-x+\text{offset}$$

Пример: makeNegativeOffset(10), вызов с 3 → 7. Функция makeNegativeOffset(offset) должна вернуть лямбду, а не сразу вычислить один ответ.

Задание 2 · Рекурсия: количество цифр

Для неотрицательного целого n не больше 10^9 рекурсивно найдите количество десятичных цифр. Если n меньше 10, ответ равен 1. Иначе уберите последнюю цифру целочисленным делением на 10.

$$D(n)=\begin{cases}1, & 0\le n<10,\\ 1+D(\lfloor n/10\rfloor), & n\ge 10.\end{cases}$$

Проверка: n = 0 → 1; n = 507 → 3.

Вариант 15 · Разность с эталоном

Задание 1 · Генератор функций

Верните функцию, которая вычитает сохранённый эталон из переданного аргумента. В отличие от расстояния, результат может быть отрицательным.

$$f(x) = x - \text{reference}$$

Пример: makeDifference(5): для 2 → −3, для 8 → 3. Функция makeDifference(reference) должна вернуть лямбду, а не сразу вычислить один ответ.

Задание 2 · Рекурсия: сумма цифр

Для неотрицательного целого n не больше 10^9 рекурсивно найдите сумму десятичных цифр. База: n = 0 даёт 0. На шаге прибавьте последнюю цифру к результату для оставшейся части числа.

$$T(0) = 0, \qquad T(n) = (n \bmod 10) + T(\lfloor n/10 \rfloor) \quad (n > 0)$$

Проверка: n = 507 → 12; n = 0 → 0.

Вариант 16+ · Две границы

Задание 1 · Генератор функций

Верните функцию ограничения x с двух сторон: ниже left — left, выше right — right, внутри интервала — x. Проверьте упорядоченность границ до создания лямбды.

$$f(x) = \min(\max(x, \text{left}), \text{right})$$

Пример: makeClamp(2, 5): для 1 → 2, для 4 → 4, для 9 → 5. Функция makeClamp(left, right) должна вернуть лямбду, а не сразу вычислить один ответ.

Задание 2 · Рекурсия: сумма от 1 до n

Для целого n от 0 до 100 рекурсивно найдите сумму чисел от 1 до n. Для n = 0 результат 0. На каждом шаге уменьшайте n на 1.

$$S(0) = 0, \qquad S(n) = n + S(n-1) \quad (n > 0)$$

Проверка: n = 4 → 10; n = 0 → 0.

Вариант 17+ · Стоимость с доплатой

Задание 1 · Генератор функций

Захватите положительную цену и неотрицательную доплату. При нулевом количестве стоимость 0; при положительном — цена товаров плюс доплата.

$$f(q) = \begin{cases} 0, & q = 0, \\ pq + c, & q > 0. \end{cases}$$

Пример: makePriceWithFee(5, 3): для 0 → 0, для 2 → 13. Функция makePriceWithFee(unitPrice, fee) должна вернуть лямбду, а не сразу вычислить один ответ.

Задание 2 · Рекурсия: факториал

Для целого n от 0 до 12 рекурсивно вычислите n!. Для n = 0 верните 1. На каждом шаге умножайте n на результат для n − 1. Результат храните в long long.

$$F(0) = 1, \qquad F(n) = nF(n-1) \quad (n > 0)$$

Проверка: n = 5 → 120; n = 0 → 1.

Вариант 18+ · Сдвиг перед рекурсией

Задание 1 · Генератор функций

Верните лямбду, принимающую целое x от −5 до 5, сдвигающую его на сохранённый shift от −5 до 5 и вычисляющую куб через рекурсивную функцию второго задания. Не вызывайте pow.

$$f(x) = (x + \text{shift})^3$$

Пример: makeShiftedPower(1), вызов с 2 → 27. Функция makeShiftedPower(shift) должна вернуть лямбду, а не сразу вычислить один ответ.

Задание 2 · Рекурсия: степень числа

Для целого основания a от −10 до 10 и целого показателя n от 0 до 8 рекурсивно вычислите a в степени n. При n = 0 возвращайте 1, в том числе для a = 0. На каждом шаге уменьшайте показатель на 1. Используйте long long.

$$P(a, 0) = 1, \qquad P(a, n) = aP(a, n-1) \quad (n > 0)$$

Проверка: a = 2, n = 5 → 32; a = −2, n = 3 → −8.

Вариант 19+ · Длина записи числа

Задание 1 · Генератор функций

Захватите ожидаемое количество цифр от 1 до 10. Возвращаемый предикат принимает неотрицательное n до 10^9 и сравнивает результат рекурсивной функции второго задания с захваченным числом.

$$f(n) = (D(n) = \text{expectedDigits})$$

Пример: makeDigitLengthCheck(3): для 507 → true, для 0 → false. Функция makeDigitLengthCheck(expectedDigits) должна вернуть лямбду, а не сразу вычислить один ответ.

Задание 2 · Рекурсия: количество цифр

Для неотрицательного целого n не больше 10^9 рекурсивно найдите количество десятичных цифр. Если n меньше 10, ответ равен 1. Иначе уберите последнюю цифру целочисленным делением на 10.

$$D(n) = \begin{cases} 1, & 0 \le n < 10, \\ 1 + D(\lfloor n/10 \rfloor), & n \ge 10. \end{cases}$$

Проверка: n = 0 → 1; n = 507 → 3.

Вариант 20+ · Сумма цифр с коэффициентом

Задание 1 · Генератор функций

Захватите целый scale от −10 до 10. Лямбда принимает неотрицательное n до 10^9, вызывает рекурсивную функцию суммы цифр из второго задания и умножает результат на сохранённый коэффициент.

$$f(n) = \text{scale} \cdot T(n)$$

Пример: makeScaledDigitSum(2), вызов с 507 → 24. Функция makeScaledDigitSum(scale) должна вернуть лямбду, а не сразу вычислить один ответ.

Задание 2 · Рекурсия: сумма цифр

Для неотрицательного целого n не больше 10^9 рекурсивно найдите сумму десятичных цифр. База: n = 0 даёт 0. На шаге прибавьте последнюю цифру к результату для оставшейся части числа.

$$T(0) = 0, \qquad T(n) = (n \bmod 10) + T(\lfloor n/10 \rfloor) \quad (n > 0)$$

Проверка: n = 507 → 12; n = 0 → 0.

Возможные вопросы к защите

Попробуйте ответить своими словами и показать соответствующее место в своей программе. Под вопросом можно открыть ориентир для ответа и перейти к нужной теме лекции.

▼ Чем вызов генератора отличается от вызова созданной функции?

Первый создаёт замыкание с настройками, второй выполняет сохранённое вычисление для переданных аргументов.

Повторить: Лекция 3 · создание функций

▼ Что запоминает захват по значению?

Покажите копию настройки внутри лямбды и объясните, почему изменение исходной переменной не меняет старое замыкание.

Повторить: Лекция 3 · создание функций

▼ Почему опасно вернуть лямбду со ссылкой на локальный параметр?

После завершения генератора параметр перестаёт существовать; ссылка не удерживает его в живых. Сравните с захватом по значению.

Повторить: Лекция 3 · создание функций · Лекция 7 · время жизни

▼ Что такое базовый случай рекурсии?

Назовите вход, для которого функция отвечает без следующего вызова, и покажите его в своём решении.

Повторить: Лекция 9 · рекурсия

▼ Почему рекурсивные вызовы доходят до базы?

Объясните уменьшение аргумента на единицу или удаление последней цифры. Покажите цепочку для небольшого числа.

Повторить: Лекция 9 · рекурсия

▼ Как связаны глубина рекурсии и стек вызовов?

Каждый незавершённый вызов сохраняет своё состояние. Объясните, почему здесь заданы небольшие ограничения, и проверьте граничный вход своего варианта.

Повторить: Лекция 9 · стек вызовов · Лекция 9 · глубина рекурсии
