Циклы

Denis Bakin

Целочисленные типы

- На первой лекции мы рассмотрели знаковые и беззнаковые целые типы
- Модификаторы размера: `long`, `short`
- Удобнее использовать фиксированные типы из `<cstdint>`

```
#include <cstdint>

int8_t a;       // 8-битный знаковый целый
uint8_t b;      // 8-битный беззнаковый целый
int16_t c;      // 16-битный знаковый целый
uint16_t d;     // 16-битный беззнаковый целый
int32_t e;      // 32-битный знаковый целый
uint32_t f;     // 32-битный беззнаковый целый
int64_t g;      // 64-битный знаковый целый
uint64_t h;     // 64-битный беззнаковый целый
```

Циклы

Определение

- Цикл — многократное повторение команд
- Состоит из:
  - тела цикла
  - условия продолжения
- Итерация — один проход тела цикла
- Внутри могут быть другие циклы, условия, инструкции

Цикл for

Общая форма

```
for (<инициализация>; <условие>; <изменение>) {
    // тело цикла
}
```

- Обычно используется, если известно количество итераций
- Переменная цикла локальна
- Условие можно задавать по-разному

Цикл for

Пример: от 1 до n

```
#include <iostream>

int main() {
    int n;
    std::cin >> n;

    for (int i = 1; i < n; ++i) {
        std::cout << i << "\n";
    }
}
```

Для `n = 5` выведется:

- `i` создается и используется только внутри цикла
- `i < n` можно заменить на `i <= n`, если необходимо включить `n`

Цикл for

Пример: от 1 до n

```cpp
#include <iostream>

int main() {
    int n;
    std::cin >> n;

    for (int i = 1; i < n; ++i) {
        std::cout << i << "\n";
    }
}
```

Для `n = 5` выведется:

1
2
3
4

- `i` создается и используется только внутри цикла
- `i < n` можно заменить на `i <= n`

Цикл for

Пример: обратный отсчет

```
#include <iostream>

int main() {
    int n;
    std::cin >> n;

    for (int i = n; i >= 0; i -= 1) {
        std::cout << i << "\n";
    }
}
```

Для `n = 5` выведется:

Цикл for

Пример: обратный отсчет

```cpp
#include <iostream>

int main() {
    int n;
    std::cin >> n;

    for (int i = n; i >= 0; i -= 1) {
        std::cout << i << "\n";
    }
}
```

Для `n = 5` выведется:

```
5
4
3
2
1
0
```

Цикл for

Задача с суммой

Посчитать: $1 + (1 + 2) + (1 + 2 + 3) + \cdots + (1 + 2 + \cdots + n)$

Цикл for

Задача с суммой

Посчитать: $1 + (1 + 2) + (1 + 2 + 3) + \cdots + (1 + 2 + \cdots + n)$

```cpp
#include <iostream>

int main() {
    uint64_t n;
    std::cin >> n;
    uint64_t sum = 0;
    uint64_t last_add = 1;

    for (uint64_t i = 2; i < n + 2; ++i) {
        sum += last_add;
        last_add += i;
    }
    std::cout << sum << '\n';
}
```

Range-based for

Итерирование по объектам

- Появился в C++11
- Позволяет пройтись по элементам контейнера
- Пример: строка

```cpp
#include <iostream>
#include <string>

int main() {
    std::string name;
    std::getline(std::cin, name);

    for (char letter: name) {
        std::cout << letter << '\t' <<
            static_cast<int>(letter) << '\n';
    }
}
```

```
Enter your name:
Nikolai
Let me spell it:
N   78
i   105
k   107
o   111
l   108
a   97
i   105
```

Цикл while

С предусловием

- Используется, если количество итераций неизвестно заранее
- Пока условие истинно — выполняем

```
#include <iostream>

int main() {
    int num;
    std::cin >> num;
    while (num) {
        std::cout << num % 10 << '\n';
        num /= 10;
    }
}
```

Цикл do-while

С постусловием

- Гарантированно выполняется хотя бы один раз
- Редко используется

```
#include <iostream>

int main() {
    int n = 1;
    do {
        std::cout << n << "\t" << n * n << "\n";
        ++n;
    } while (n <= 10);
}
```

Break и continue

- `break` — прерывает цикл
- `continue` — пропускает итерацию и переходит к следующей

Пример: бесконечный калькулятор с выходом по условию

```
#include <cstdint>
#include <iostream>

int main() {
    int64_t a, b;
    char operation;
    while (true) {
        std::cin >> a >> operation >> b;
        if (!a && !b && operation == '0') {
            break;
        }
        if ((operation == '/' || operation == ':') && !b) {
            continue;
        }
        // вычисления ...
    }
}
```

Вложенные циклы

Таблица умножения

```
#include <iostream>

int main() {
    for (int i = 1; i <= 10; ++i) {
        for (int j = 1; j <= 10; ++j) {
            std::cout << i  j << "\t";
        }
        std::cout << "\n";
    }
}
```

Вложенные циклы

Таблица умножения

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

Типовые приемы

Счетчик

- Подсчет элементов, удовлетворяющих условию

```
#include <iostream>

int main() {
    int new_num;
    std::cin >> new_num;

    int cnt = 0;
    while (new_num) {
        if (new_num % 2 == 0) {
           ++cnt;
        }
        std::cin >> new_num;
    }
    std::cout << cnt << std::endl;
}
```

Типовые приемы

Поиск минимума и максимума

```
#include <iostream>
#include <algorithm>

int main() {
    int new_num;
    std::cin >> new_num;
    int max = new_num, min = new_num;

    while (new_num) {
        max = std::max(max, new_num);
        min = std::min(min, new_num);
        std::cin >> new_num;
    }

    printf("%d %d\n", min, max);
}
```

Типовые ошибки

Бесконечный цикл

- Забыли изменить переменную цикла
- Условие всегда истинно
- Часто проявляется в `while`

```
#include <iostream>

int main() {
    int i = 0;
    while (i < 10) {
        std::cout << i << "\n";
        // забыли i++
    }
}
```

Типовые ошибки

Ошибка границ

- Использование <= вместо <
- Вылет за пределы массива
- Печать лишнего элемента

```cpp
#include <iostream>

int main() {
    int n = 5;
    for (int i = 0; i <= n; ++i) {
        std::cout << i << "\n"; // выведет 0..5, а не 0..4
    }
}
```

Типовые ошибки

Использование неинициализированной переменной

- Переменная цикла создаётся внутри `for`
- После цикла она недоступна

```
#include <iostream>

int main() {
    for (int i = 0; i < 5; ++i) {
        std::cout << i << " ";
    }
    std::cout << i; // ошибка: i не существует
}
```

Типовые ошибки

Нарушение условий выхода

- Сложные условия → легко ошибиться
- Ошибка логического оператора (&& ☒ ||)

```
#include <iostream>

int main() {
    int n;
    std::cin >> n;

    while (n < 0 && n > 100) { // условие всегда ложно
        std::cout << "Inside loop\n";
    }
}
```

Типовые ошибки

Ошибка при использовании break/continue

- `continue` может “перепрыгнуть” важный код
- `break` выходит только из одного цикла

```cpp
#include <iostream>

int main() {
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            if (j == 1)
                break; // прервёт только внутренний цикл
            std::cout << i << " " << j << "\n";
        }
    }
}
```