Условия, ветвление. Циклы

Denis Bakin

Представление целых чисел в памяти компьютера

Степени двойки

- Удобно запоминать степени 2 в двоичной системе

Примеры: 255, 1023, 2048, 8.

Представление целых чисел в памяти компьютера

Степени двойки

- Удобно запоминать степени 2 в двоичной системе
- $2^n = ?$

Примеры: 255, 1023, 2048, 8.

Представление целых чисел в памяти компьютера

Степени двойки

- Удобно запоминать степени 2 в двоичной системе
- $2^n = ?$
- $2^n = 1\underbrace{0 \ldots 0}_{n}$

Примеры: 255, 1023, 2048, 8.

Представление целых чисел в памяти компьютера

Степени двойки

- Удобно запоминать степени 2 в двоичной системе
- $2^n = ?$
- $2^n = 1\underbrace{0\ldots0}_{n}$
- $2^n - 1 = ?$

Примеры: 255, 1023, 2048, 8.

Представление целых чисел в памяти компьютера

Степени двойки

- Удобно запоминать степени 2 в двоичной системе
- $2^n = ?$
- $2^n = 1\underbrace{0 \ldots 0}_{n}$
- $2^n - 1 = ?$
- $2^n - 1 = \underbrace{1 \ldots 1}_{n}$

Примеры: 255, 1023, 2048, 8.

Представление целых чисел в памяти компьютера

Sign-magnitude

- Первый бит — знак
- Остальные биты — модуль
- Арифметика неудобна для компьютера
- Пример: -6 в 4-битном формате $\rightarrow$ 1110

Бинарное значение	Sign-magnitude	Беззнаковое
00000000	0	0
00000001	1	1
...	...	...
01111111	127	127
10000000	-0	128
10000001	-1	129
...	...	...
11111111	-127	255

Двоично-дополнительный код

Преобразование

- Универсальное представление отрицательных чисел
- Алгоритм:
  1. Записать модуль в двоичной системе
  2. Инвертировать все биты
  3. Прибавить 1
- Пример: -6 в 4-битном формате $\rightarrow$ 1010

Двоично-дополнительный код

Проверка примера -6

Бит	1	0	1	0
Значение	-8	4	2	1

$1010 = -8 + 2 = -6$

То есть первый бит отвечает за -2^{n-1}, а остальные биты — за положительные степени двойки.

Двоично-дополнительный код

Таблица значений

Биты	Десятичное значение
0000	0
0001	1
0010	2
...	...
0111	7
1000	−8
1001	−7
...	...
1110	−2
1111	−1

Двоично-дополнительный код

Преимущества

- Нет отдельной логики для знака
- Арифметика идентична беззнаковой
- Переполнение $\rightarrow$ “зацикливание”

Биты	Десятичное значение	Комментарий
10000000	-128	
10000001	-127	
...	...	
11111111	-1	$11111111_2 + 1_2 = 100000000_2$. Записываем 8 правых бит
00000000	0	
00000001	1	
...	...	
01111111	127	
10000000	-128	“Зацикливание”, то есть произошло “переполнение”, “overflow”
...	...	

Переполнение целых типов

- Слишком большое число → берутся правые биты
- По сути: остаток при делении на $2^{\text{размер типа}}$
- Ошибки не возникает — программа продолжает выполнение

```cpp
#include <iostream>

int main() {
    unsigned char c_1 = 0;
    c_1 -= 1; // underflow
    std::cout << (int)c_1 << '\n'; // 255

    c_1 = 300;
    std::cout << (int)c_1 << '\n'; // 44
}
```

Условия

Логические операции

- `!A` — отрицание (НЕ)

Условия

Логические операции

- `!A` — отрицание (НЕ)
- `A && B` — конъюнкция (И)

Условия

Логические операции

- `!A` — отрицание (НЕ)
- `A && B` — конъюнкция (И)
- `A || B` — дизъюнкция (ИЛИ)

Условия

Множественные ветвления

```
if (condition1) {
    // ...
} else if (condition2) {
    // ...
} else {
    // ...
}
```

Условия

Пример: билет в кино

- Условие для клиента: номер не делится на 6 и оканчивается четной цифрой
- Сотрудники: номера < 10
- Директор: билет 777

Условия

Пример: билет в кино

- Условие для клиента: номер не делится на 6 и оканчивается четной цифрой
- Сотрудники: номера < 10
- Директор: билет 777

```cpp
#include <iostream>

int main() {
    int ticket_id;
    std::cin >> ticket_id;
    if (ticket_id % 6 != 0 && (ticket_id % 10) % 2 == 0) {
        printf("Enjoy the movie, customer!\n");
    } else if (ticket_id >= 0 && ticket_id < 10) {
        printf("Hi, employee\n");
    } else if (ticket_id == 777) {
        printf("Hello, CEO!\n");
    } else {
        printf("Invalid ticket ID!\n");
    }
}
```

Условия

С числами с плавающей точкой

- Прямое сравнение == может быть неверным
- Использовать сравнение с погрешностью

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

int main() {
    float number;
    std::cin >> number;

    if (std::abs(number - 3.0) < 0.000001) {
        std::cout << "you have entered 3.0\n";
    }
}
```

Условия

Switch-case

- Удобен для проверки на равенство множеству значений
- Работает только с элементарными типами (числа, символы)

```
switch (operation) {
    case '+':
        result = a + b;
        break;
    case '-':
        result = a - b;
        break;
    default:
        result = 0;
}
```

Условия

Пример калькулятора

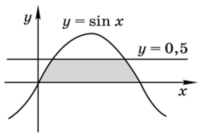
```cpp
#include <cstdint>
#include <iostream>

int main() {
    int64_t a, b;
    char operation;
    std::cin >> a >> operation >> b;

    int64_t result;
    switch (operation) {
        case '+': result = a + b; break;
        case '-': result = a - b; break;
        case '*': result = a * b; break;
        case '/': result = a / b; break;
        case '%': result = a % b; break;
        default: result = 0;
    }

    std::cout << result << "\n";
}
```

Условия

Геометрическая задача

Задача №112166. Точка - 2

Напишите программу, которая определяет, попала ли точка с заданными координатами в заштрихованную область.



Входные данные

Входная строка содержит два вещественных числа – координаты точки на плоскости (сначала x -координата, затем – y -координата).

Выходные данные

Программа должна вывести слово 'YES', если точка попала в заштрихованную область, и слово 'NO', если не попала.

Figure 1: Задача 112166 на использование условий

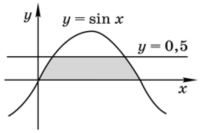
Условие области:

Условия

Геометрическая задача

Задача №112166. Точка - 2

Напишите программу, которая определяет, попала ли точка с заданными координатами в заштрихованную область.



Входные данные

Входная строка содержит два вещественных числа – координаты точки на плоскости (сначала x -координата, затем – y -координата).

Выходные данные

Программа должна вывести слово 'YES', если точка попала в заштрихованную область, и слово 'NO', если не попала.

Figure 2: Задача 112166 на использование условий

Условие области:

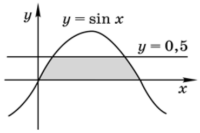
- $y \leq \sin(x)$

Условия

Геометрическая задача

Задача №112166. Точка - 2

Напишите программу, которая определяет, попала ли точка с заданными координатами в заштрихованную область.



Входные данные

Входная строка содержит два вещественных числа – координаты точки на плоскости (сначала x -координата, затем – y -координата).

Выходные данные

Программа должна вывести слово 'YES', если точка попала в заштрихованную область, и слово 'NO', если не попала.

Figure 2: Задача 112166 на использование условий

Условие области:

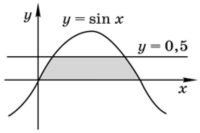
- $y \leq \sin(x)$
- $y \leq 0.5$

Условия

Геометрическая задача

Задача №112166. Точка - 2

Напишите программу, которая определяет, попала ли точка с заданными координатами в заштрихованную область.



Входные данные

Входная строка содержит два вещественных числа – координаты точки на плоскости (сначала x -координата, затем – y -координата).

Выходные данные

Программа должна вывести слово 'YES', если точка попала в заштрихованную область, и слово 'NO', если не попала.

Figure 2: Задача 112166 на использование условий

Условие области:

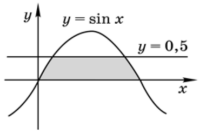
- $y \leq \sin(x)$
- $y \leq 0.5$
- $y \geq 0$

Условия

Геометрическая задача

Задача №112166. Точка - 2

Напишите программу, которая определяет, попала ли точка с заданными координатами в заштрихованную область.



Входные данные

Входная строка содержит два вещественных числа – координаты точки на плоскости (сначала x -координата, затем – y -координата).

Выходные данные

Программа должна вывести слово 'YES', если точка попала в заштрихованную область, и слово 'NO', если не попала.

Figure 2: Задача 112166 на использование условий

Условие области:

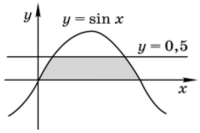
- $y \leq \sin(x)$
- $y \leq 0.5$
- $y \geq 0$
- $x \geq 0$

Условия

Геометрическая задача

Задача №112166. Точка - 2

Напишите программу, которая определяет, попала ли точка с заданными координатами в заштрихованную область.



Входные данные

Входная строка содержит два вещественных числа – координаты точки на плоскости (сначала x -координата, затем – y -координата).

Выходные данные

Программа должна вывести слово 'YES', если точка попала в заштрихованную область, и слово 'NO', если не попала.

Figure 2: Задача 112166 на использование условий

Условие области:

- $y \leq \sin(x)$
- $y \leq 0.5$
- $y \geq 0$
- $x \geq 0$
- $x \leq \pi$

Условия

Решение задачи

```
#include <iostream>
#include <cmath>

int main() {
    float x, y;
    std::cin >> x >> y;
    if (y <= std::sin(x) && y >= 0 && y <= 0.5 && x >= 0 && x <= M_PI) {
        printf("YES\n");
    } else {
        printf("NO\n");
    }
}
```

Отладка программ

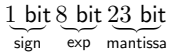
Основные методы

- Разбор случаев “на бумаге”
- Трассировка (дополнительный вывод)
- Использование отладчика

Представление чисел с плавающей точкой

Формат IEEE 754 (float)

- 32 бита:
  - 1 бит — знак
  - 8 бит — экспонента
  - 23 бита — мантисса



Представление чисел с плавающей точкой

Компоненты

- `sign`: 0 — положительное, 1 — отрицательное
- `exp`: показатель степени, хранится со смещением 127
- `mantissa`: значащие биты дробной части

Представление чисел с плавающей точкой

Интерпретация

- $0 \leq exp \leq 254$: нормализованное число $(-1)^{sign} \times 2^{exp-127} \times 1.[mantissa]$
- $exp = 0$: денормализованное число $(-1)^{sign} \times 2^{-126} \times 0.[mantissa]$
- $exp = 255, mantissa = 0$: $\pm\infty$
- $exp = 255, mantissa \neq 0$: `NaN` (“не число”)

Представление чисел с плавающей точкой

Интересный факт

- Чем ближе число к нулю → тем плотнее значения
- При росте экспоненты шаг между числами увеличивается
- Денормализованные числа: равномерный шаг
- Нормализованные: шаг удваивается при увеличении exp

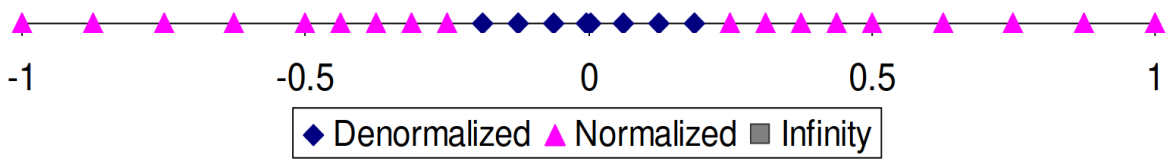


Figure 3: Распределение чисел с плавающей точкой

Представление чисел с плавающей точкой

Тип double

- Аналогично float, но 64 бита
  - 1 бит — знак
  - 11 бит — экспонента
  - 52 бита — мантисса
- Более высокая точность
- Более широкий диапазон значений

Представление чисел с плавающей точкой

Вопросы для размышления

- Почему нельзя точно записать любое рациональное число?
- Как доступная точность зависит от модуля числа?
- Сколько есть способов записать 0?
- Как перемножить два `float` на уровне битов?

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";
        }
    }
}
```