ЛЕКЦИЯ 06
УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

Условный оператор `if`

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

Условный оператор `if`

```
if (expression) {
      // какой-то код
}
```

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

Условный оператор `if`

```
if (expression) {
      // какой-то код
}
// другой код
```

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

Условный оператор `if`

```
if (expression) {
      // какой-то код
} else {
// другой код
}
```

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

Условный оператор `if`

```
if (expression) {
      // какой-то код
} else {
// другой код
}
// ещё код
```

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

Условный оператор `if`

```
if (expression) {
      // какой-то код
} else if (expression) {
// другой код
}
// ещё код
```

УПРАЖНЕНИЯ

Условный оператор `if`

```
int x = 10;
int y = 20;
int z = 30;
if (x % 2 == 0) {
      std::cout << “x is even” << std::endl;
}
std::cout << “x is odd” << std::endl;
```

Что будет на экране?

УПРАЖНЕНИЯ

Условный оператор `if`

```
int x = 10;
int y = 20;
int z = 30;
if (x % 2 == 0) {
      std::cout << “x is even” << std::endl;
} else
std::cout << “x is odd” << std::endl;
```

Что будет на экране?

УПРАЖНЕНИЯ

Условный оператор `if`

```
int x = 10;
int y = 20;
int z = 30;
if (x % 2 == 0) {
      std::cout << “x is even” << std::endl;
} else if (z & 1)
std::cout << “x and z is odd” << std::endl;
```

Что будет на экране?

УПРАЖНЕНИЯ

Условный оператор `if`

```
int x = 10;
int y = 20;
int z = 30;
if (x + y - z) {
      std::cout << 1 << std::endl;
} else if (x - y + z)
std::cout << 2 << std::endl;
std::cout << 3 << std::endl;
```

Что будет на экране?

УПРАЖНЕНИЯ

Условный оператор `if`

```
int x = 10;
int y = x = 20;
int z = y = 30;
if (x + y - z) {
      std::cout << 1 << std::endl;
} else if (x - y + z)
std::cout << 2 << std::endl;
std::cout << 3 << std::endl;
```

Что будет на экране?

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

Цикл `while`

```
while (expression) {
      // какой-то код
}
// другой код
```

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

Цикл `while`

```
int num = 1;
while (num < 10) {
      num *= 2;
}
std::cout << num;
```

Что будет на экране?

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

Цикл `while`

```
int num = 10;
while (num < 10) {
      num *= 2;
}
std::cout << num;
```

Что будет на экране?

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

Цикл do-while

```
do {
        // какой-то код
} while (expression)
// другой код
```

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

Цикл do-while

```
int num = 1;
do {
        num *= 2;
} while (num < 10)
std::cout << num;
```

Что будет на экране?

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

Цикл do-while

```
int num = 10;
do {
        num *= 2;
} while (num < 10)
std::cout << num;
```

Что будет на экране?

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

Цикл `for`

```
for {initialization; condition; iter-expression}
      // какой-то код
}
// другой код
```

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

Цикл `for`

```
int num = 0
for {int i = 1; i < 10; i *= 2}
      num += 1;
}
std::cout << num;
```

Что будет на экране?

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

Цикл for-ever

```
for {;;}
      //какой-то код
}
//какой-то код
```

Что будет на экране?

ОСНОВНЫЕ УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

Ключевые слова `break` и `continue`.

`break` – прервать выполнение цикла и продолжить выполнение кода после тела цикла.

`continue` – перейти к следующей итерации цикла

КОРОТКО О ЦИКЛАХ

В языке C++ есть три основных типа циклов: `for`, `while` и `do-while`

Циклы `while` выполняются пока какое-то условие истинно.

```
while (i < 100) { /* тело цикла */}
```

Циклы `do-while`: точно выполняются однажды

```
do { /* тело цикла */} while (i < 100) ;
```

Циклы `for`: содержат инициализацию и изменение **индуктивной переменной**

```
for (i = 0; i < 100; ++i) { /* тело цикла */}
i = 0; while(i < 100) { /* тело цикла */; ++i }
```

КОРОТКО О ФУНКЦИЯХ

Объявление функции задает её сигнатуру (типы аргументов и тип значения).

```
int foo(int x, double y);
void bar(); // функция не возвращает значение
```

Функция может быть вызвана даже если она еще не определена

```
int t = 42; int s; s = foo(t, 1.0);
```

Где-то в программе (но только один раз) должно найтись тело функции

```
int foo(int x, double y) { int t = x + (int) y; return t;}
```

Возвращаемое значение можно проигнорировать, даже если оно не `void`

```
int t = 42; foo(t, 1.0); // ok
```

РУССКО-КРЕСТЬЯНСКОЕ УМНОЖЕНИЕ

```
int rp_mult(int a, int b, int mod) {
    if (b == 1) return a % mod;
    if (b == 0) return 0;
    int res = 0;
    while (b != 0) {
        if (b & 1) res += a % mod;
        a = ((a % mod) + (a % mod)) % mod;
        b >>= 1;
    }
    return res % mod;
}
```

НАИБОЛЬШИЙ ОБЩИЙ ДЕЛИТЕЛЬ

Говорят, что целое число a делит b, если их частное $\frac{a}{b}$ является целым числом. $a\backslash \mathrm{b} \Leftrightarrow \exists c \mid b = ac$

Наибольшее среди чисел, которые делят оба числа x и y называется их наибольшим общим делителем (greatest common divisor, gcd).

$$a = \gcd(x, y) \Leftrightarrow a = \max\{n \mid (n\backslash x) \wedge (n\backslash \mathrm{y})\}$$

Например наибольшим общим делителем чисел 14 и 8 является 2

Что является наибольшим общим делителем чисел -14 и 8?

Как найти наибольший общий делитель чисел 893623 и 3182456?

ПИШЕМ АЛГОРИТМ

```
int gcd(int x, int y) {
        int min_num = x > y ? x : y;
        int result = 1;
        for (int i = 1; i <= min_num; ++i)
                if (x % i == 0 && y % i == 0)
                        result = i;
        return result;
}
```

НЕМНОГО МАТЕМАТИКИ

$x = y \cdot p + q$

Пусть k – общий делитель x и y.

Тогда $q = x - y \cdot p$ тоже делится на k

Значит $\gcd(x, y) = \gcd(y, q)$

Например посчитаем $\gcd(35, 13)$

$35 = 13 \cdot 2 + 9$ значит $\gcd(35,13) = \gcd(13,9)$

$13 = 9 \cdot 1 + 4$ значит $\gcd(13,9) = \gcd(9,4)$

$9 = 4 \cdot 2 + 1$ значит $\gcd(9,4) = \gcd(4,1) = 1$

ПИШЕМ АЛГОРИТМ

$a \backslash b \Leftrightarrow \exists c \mid b = ac$

$a = \gcd(x, y) \Leftrightarrow a = \max\{n \mid (n \backslash x) \wedge (n \backslash y)\}$

```
int gcd(int x, int y) {
// Что напишете?
}
```

$x = y \cdot p + q$
Пусть k – общий делитель x и y.
Тогда $q = x - y \cdot p$ тоже делится на k
Значит $\gcd(x, y) = \gcd(y, q)$

Посчитайте `gcd(893623, 3182456)`
Как будете тестировать функцию?

ПИШЕМ АЛГОРИТМ

$a \backslash b \Leftrightarrow \exists c \mid b = ac$

$a = \gcd(x, y) \Leftrightarrow a = \max\{n \mid (n \backslash x) \wedge (n \backslash y)\}$

```
int gcd(int x, int y) {
        int q;
        assert(y != 0);
        q = x % y;
        if (q == 0) return y;
        return gcd(y, q);
}
```

Посчитайте `gcd(893623, 3182456)`

Как будете тестировать функцию?

$x = y \cdot p + q$

Пусть k – общий делитель x и y.

Тогда $q = x - y \cdot p$ тоже делится на k

Значит $\gcd(x, y) = \gcd(y, q)$

ВНЕЗАПНАЯ ПРОБЛЕМА

Мы считаем $\gcd(14, -8)$ и внезапно оказывается, что это -2

Все дело в тонкой разнице

В математике Евклидово деление определено следующим образом: $a = q \cdot b + r, 0 \leq r < |b|$

Тогда как в языках C и C++ операция % ведет себя немного иначе

```
assert(a == (a / b) * b + (a % b));
```

Чтобы получить настоящий алгоритм Евклида, надо реализовать настоящее Евклидово деление.

ЕВКЛИДОВО ДЕЛЕНИЕ И НОД

```
int eq_mod(int x, int y) {
        int q = x % y;
        if (q < 0) q += abs(y);
        return q;
}
int gcd(int x, int y) {
        int q;
        assert(y != 0);
        q = eq_mod(x, y);
        if (q == 0) return y;
        return gcd(y, q);
}
```

ДОМАШНЕЕ ЗАДАНИЕ

1. Написать функцию русско-крестьянского возведения в степень по модулю
2. Написать функцию нахождения наименьшего общего кратного
3. Изобразить блок-схемы для ваших функций

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Дорохова Т.Ю., Основы алгоритмизации и программирования : учебное пособие для СПО / Т.Ю. Дорохова, И.Е. Ильина. – Саратов, Москва : Профобразование, Ай, Пи Ар Медиа, 2022. – 139 с.
2. Кудинов Ю.И., Основы алгоритмизации и программирования : учебное пособие для СПО / Ю.И. Кудинов, А.Ю. Келина. – 2-е изд. – Липецк, Саратов: Липецкий государственный технический университет, Профообразование, 2020. – 71 с.
3. Дональд Кнут, Искусство программирования. Том 1. Основные алгоритмы / Ю.В. Козаченко. - 3-е изд – Москва, Санкт-Петербург: ВИЛЬЯМС, 2018. – 721 с.