09.09.2024

Данные в C. Ввод-вывод. Операции и операторы. Циклы и условия.

Филиппов Михаил Витальевич

m.filippov@g.nsu.ru

89232283872

Императивное программирование, 2024-2025

Новосибирский государственный университет

*НАСТОЯЩАЯ НАУКА

Давайте познакомимся

Филиппов Михаил Витальевич

- Окончил магистратуру ФФ НГУ
- Окончил аспирантуру ИТ СО РАН
- Являюсь м.н.с. ИТ СО РАН
- 7+ лет опыт в программировании C/C++

Адженда

Введение в данные в С	Ввод-вывод начальные сведения	Операции и операторы	Циклы и условия
15 минут	10 минут	25 минут	40 минут

Адженда

Введение в данные в С	Ввод-вывод начальные сведения	Операции и операторы	Циклы и условия
15 минут	10 минут	25 минут	40 минут

Демонстрационная программа

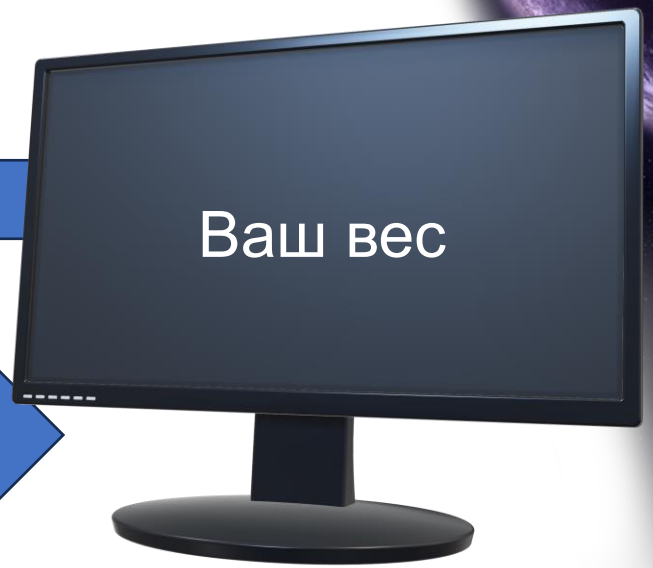
```c
/* platinum, c -- ваш вес в платиновом эквиваленте */
#include <stdio.h>
int main(void)
{
    float weight; /* вес пользователя */
    float value;  /* платиновый эквивалент */
    printf("Хотите узнать свой вес в платиновом эквиваленте?\n");
    printf("Давайте подсчитаем.\n");
    printf("Пожалуйста, введите свой вес, выраженный в фунтах: ");
    /* получить входные данные от пользователя */
    scanf("%f", &weight);
    /* считаем, что цена родия равна $1700 за тройскую унцию */
    /* 14.5833 коэффициент для перевода веса, выраженного в фунтах, в тройские унции */
    value = 1700.0 * weight * 14.5833;
    printf("Ваш вес в платиновом эквиваленте составляет $%.2f.\n", value);
    printf("Вы легко можете стать достойным этого! Если цена платины падает, \n");
    printf("ешьте больше для поддержания своей стоимости.\n");
    return 0;
}
```

printf() и scanf()

```c
/* platinum, c*/
.
.
int main(void)
{
    .
    .
    .
    scanf("%f", &weight);
    .
    .
    .
    printf("Ваш вес …", value);
    .
    .
    .
    return 0;
}
```

Получение ввода с клавиатуры

Отображение вывода программы



Ключевые слова для типов данных

Ключевые слова в исходном стандарте K&R C	Ключевые слова, добавленные стандартом C90	Ключевые слова, добавленные стандартом C99
int	signed	_Bool
long	void	_Complex
short		_Imaginary
unsigned		
char		
float		
double		

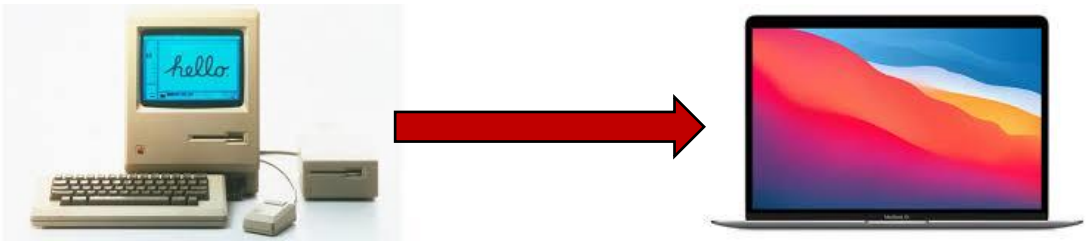
Ключевые слова для типов данных

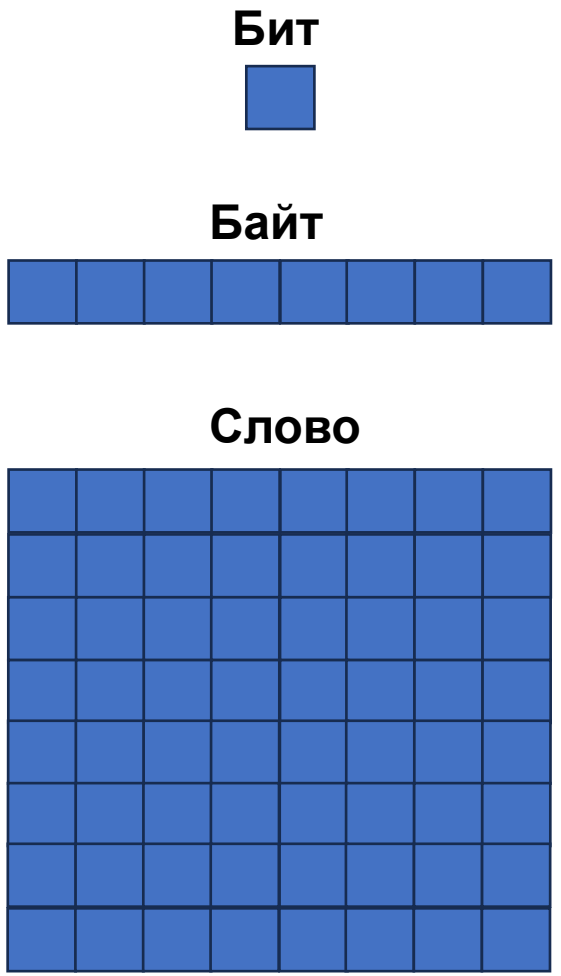
- **int** – основной класс целых чисел, применяемых в С
- **long, short, unsigned, signed** – для указания вариаций базового типа
- **char** – символьные данные, к которым относятся буквы алфавита и другие символы, такие как #, $, % и *. Тип данных **char** можно также применять для представления небольших целых чисел
- **float, double** – представление чисел с плавающей запятой
- **_Bool** используется для булевских значений (true и false)
- **_Complex** и **_Imaginary** – комплексные и мнимые числа

```c
int num = 100;
unsigned int short num_s = 11;
long long int num_l = 10000000000;
char c = 'c';
char c = 10;
long double num_d = 1.2345;
_Bool logic = true; //1
double _Complex num_c = 3 + 4*I;
double _Imaginary num_i = 4*I;
```

Биты, байты и слова

Бит – минимальная единица памяти. Может иметь значение 0 или 1. _Boolean имеет размер 1 бит
Байт — это наиболее часто используемая единица памяти компьютера. Практически на всех машинах байт состоит из 8 битов, и это является стандартным определением байта. Поскольку бит может принимать значение 0 или 1, байт обеспечивает 256 (т.е. 2^8) возможных комбинаций нулей и единиц.
Слово — это естественная единица памяти для компьютера конкретного типа, зависит от разрядности системы. В настоящее время используются 64-битные слова. Большие размеры слова позволяют быстрее передавать данные и делают доступным больший объем памяти.





Объявление переменных

```c
int pigs;
pigs = 24;
// pigs = 24
int hogs = 21;
// hogs = 21
int cows = 32, goats = 14;
// cows = 32, goats = 14
int dogs, cats = 94; /* допустимая, но неудачная форма */
// dogs - не инициализирована, cats = 94
```

Всегда нужно стараться инициализировать переменные!

int занимает ***4 байта*** и предназначен ***для хранения целых чисел.***
Диапазон чисел ***от -2 147 483 648 до 2 147 483 647***

Целочисленные типы

- Тип ***short int***, или ***short***, может использовать ***меньший объем памяти***, чем int, и тем самым экономить память в случае, когда требуются только небольшие числа.
- Тип ***long int***, или ***long,*** может занимать ***больший объем памяти***, чем int, позволяя представлять крупные целочисленные значения.
- Тип ***long long int***, или ***long long*** (введен стандартом C99), может занимать ***больше памяти, чем long***. Для этого типа используются ***минимум 64 бита***.
- Тип ***unsigned int***, или ***unsigned***, применяется для переменных, которые принимают только неотрицательные значения. Этот тип сдвигает диапазон хранимых чисел. Бит, который использовался для представления знака, теперь становится еще одной двоичной цифрой, делая возможным представление большего числа.
- Типы ***unsigned long int***, или ***unsigned long***, и ***unsigned short int***, или ***unsigned short***, распознаются как ***допустимые стандартом C90***. В ***стандарте C99*** к ним добавлен тип ***unsigned long long int***, или ***unsigned long long***.
- Ключевое слово ***signed*** может применяться с любыми типами со знаком, ***чтобы явно указать свое намерение***. Например, ***short, short int, signed short*** и ***signed short int*** являются ***именами одного и того же типа***.

Целочисленное переполнение

```c
/* toobig.c -- превышение максимально возможного значения int в системе */
#include <stdio.h>
int main(void)
{
    int i = 2147483647;
    unsigned int j = 4294967295;
    printf("%d %d %d\n", i, i + 1, i + 2);
    printf("%u %u %u\n", j, j + 1, j + 2);
    return 0;
}
//2147483647 -2147483648 -2147483647
//4294967295 0 1
```

Константы long и long long

long – дописывается буква l или L
unsigned – дописывается буква u или U

!!! Рекомендуется использовать заглавные буквы для наглядности !!!

```
#include <stdio.h>
int main()
{
    unsigned long long B = 875116400981445255 2ull; // не рекомендуется
    unsigned long long A = 875116400981445255 2ULL;
    long C = 875116400981445255 2L;
    unsigned long B = 875116400981445255 2UL;
    long long A = 875116400981445255 2LL;
}
```

Адженда

Введение в данные в С	Ввод-вывод начальные сведения	Операции и операторы	Циклы и условия
15 минут	10 минут	25 минут	40 мин

Ввод-вывод целочисленных типов

```
#include <stdio.h>
int main (void)
{
    unsigned int un = 3000000000; /* система с 32-битным типом
int */
    short end = 200; /* и 16-битным типом short */
    long big = 65537;
    long long verybig = 12345678908642;
    printf("un = %u, но не %d\n", un, un);
    printf("end = %hd и %d\n", end, end);
    printf("big = %ld, но не %hd\n", big, big);
    printf ("verybig = %lld, но не %ld\n", verybig, verybig);
    return 0;
}
//un = 3000000000, но не -1294967296
//end = 200 и 200
//big = 65537, но не 1
//verybig = 12345678908642, но не 1942899938
```

int – спецификатор %d
unsigned int – спецификатор %u
long – спецификатор %ld
unsigned long - спецификатор %lu
short - спецификатор %hd
long long - спецификатор %lld
unsigned long - спецификатор %llu

Ввод-вывод целочисленных типов

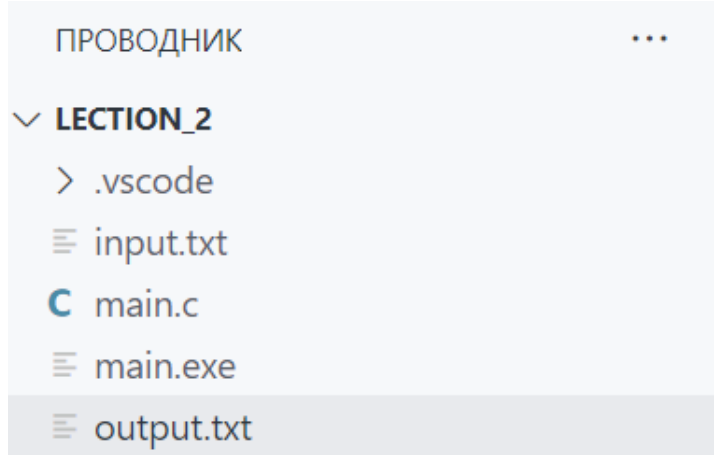
```c
#include <stdio.h>
int main()
{
    int age;       // переменная
    printf("Введите информацию о своем возрасте.\n");
    scanf("%d", &age); // здесь должен быть указан символ &
    printf("%d\n", age);
    return 0;
}
// Введите информацию о своем возрасте.
// 27
// 27
```

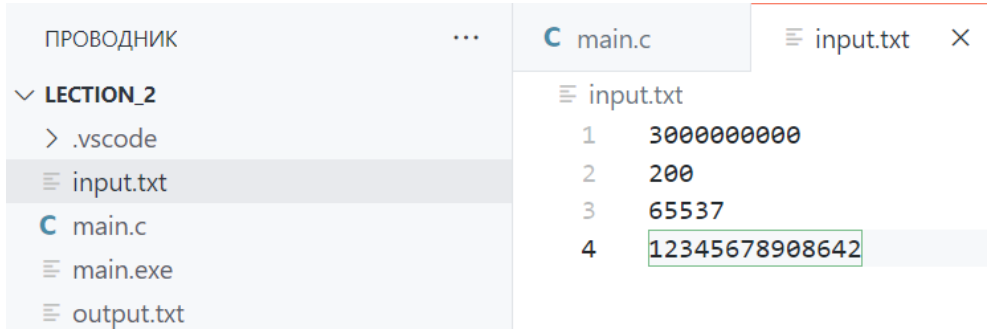
int – спецификатор %d
unsigned int – спецификатор %u
long – спецификатор %ld
unsigned long - спецификатор %lu
short - спецификатор %hd
long long - спецификатор %lld
unsigned long - спецификатор %llu

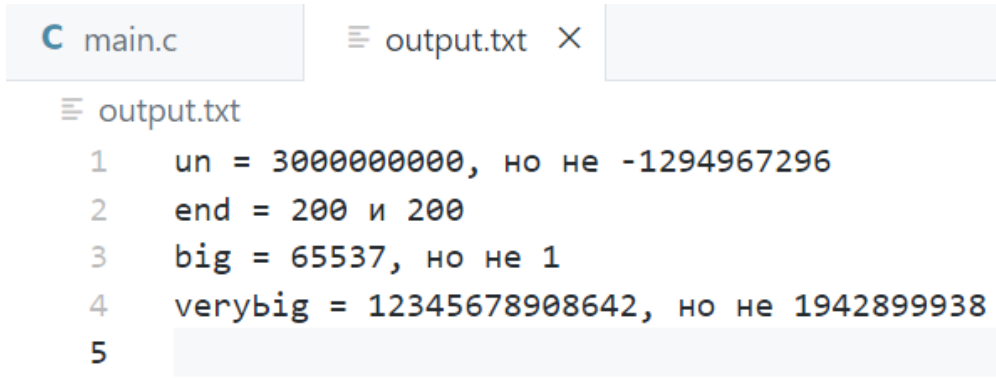
- Если вы используете функцию scanf (), чтобы прочитать значение для переменной одного из базовых типов, предварите имя переменной символом &.

Ввод вывод в файл

```c
#include <stdio.h>
int main(void)
{
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    unsigned int un;
    short end;
    long big;
    long long verybig;
    scanf("%u %hd %ld %lld", &un, &end, &big, &verybig);
    printf("un = %u, но не %d\n", un, un);
    printf("end = %hd и %d\n", end, end);
    printf("big = %ld, но не %hd\n", big, big);
    printf("verybig = %lld, но не %ld\n", verybig, verybig);
    fclose(stdin);
    fclose(stdout);
    return 0;
}
// un = 3000000000, но не -1294967296
// end = 200 и 200
// big = 65537, но не 1
// verybig = 12345678908642, но не 1942899938
```







Адженда

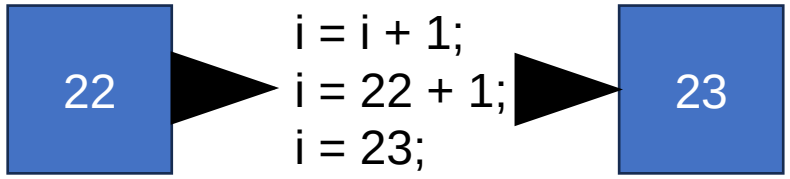
Введение в данные в С	Ввод-вывод начальные сведения	Операции и операторы	Циклы и условия
15 минут	10 минут	25 минут	40 минут

Операция присваивания: =

bmw = 2002;

Элемент, расположенный слева от знака =, представляет собой имя переменной, а элемент справа — значение, присваиваемое этой переменной. Символ = называется операций присваивания.

~~**2002 = bmw;**~~




```
#include <stdio.h>
int main(void)
{
    system("chcp 65001");
    int jane, tarzan, cheeta;
    cheeta = tarzan = jane = 68;
    printf("                    чита    тарэан  джейн\n");
    printf("Счет первого раунда %4d %8d %8d\n", cheeta,
tarzan, jane);
    return 0;
}
//                    чита     тарэан    джейн
// Счет первого раунда   68       68       68
```

Объекты данных, l-значения, r-значения и операнды

- Объект данных — это общий термин для обозначения области хранения данных, которая может применяться для удержания значений.
- Термин l-значение в C применяется для обозначения любого имени или выражения подобного рода, идентифицирующего конкретный объект данных. Объект относится к фактической области хранения данных, но 1-значение — это метка, которая используется для определения местоположения этой области памяти.
- Модифицируемое l-значение – термин для идентификации объекта, чье значение может изменяться. Например, левая часть оператора присваивания должна быть модифицируемым l-значением. В современном стандарте предлагается более точный термин: ***значение локатора объекта***.
- Термин r-значение относится к величинам, которые могут быть присвоены модифицируемым l-значениям, но которые сами не являются значениями. R-значениями могут быть константы, переменные или любое другое выражение, которое в результате дает значение, например, вызов функции.

```
int ex;
int why;
int zee;
const int TWO=2;
why=42;
zee=why;
ex=TWO*(why+zee);
```

Ввод-вывод целочисленных типов

```
#include <stdio.h>
int main(void)
{
    system("chcp 65001");
    int jane, tarzan, cheeta;
    cheeta = tarzan = jane = 68;
    printf("                      чита    тарэан  джейн\n");
    printf("Счет первого раунда %4d %8d %8d\n", cheeta,
tarzan, jane);
    return 0;
}
//                    чита    тарэан  джейн
// Счет первого раунда   68       68       68
```

int – спецификатор %d
unsigned int – спецификатор %u
long – спецификатор %ld
unsigned long - спецификатор %lu
short - спецификатор %hd
long long - спецификатор %lld
unsigned long - спецификатор %llu

- Если вы используете функцию scanf (), чтобы прочитать значение для переменной одного из базовых типов, предварите имя переменной символом &.

Операция сложения: +
Операция вычитания: -

- Операция сложения приводит к суммированию двух значений с обеих сторон знака +. Суммируемые значения (операнды) могут быть как переменными, так и константами.

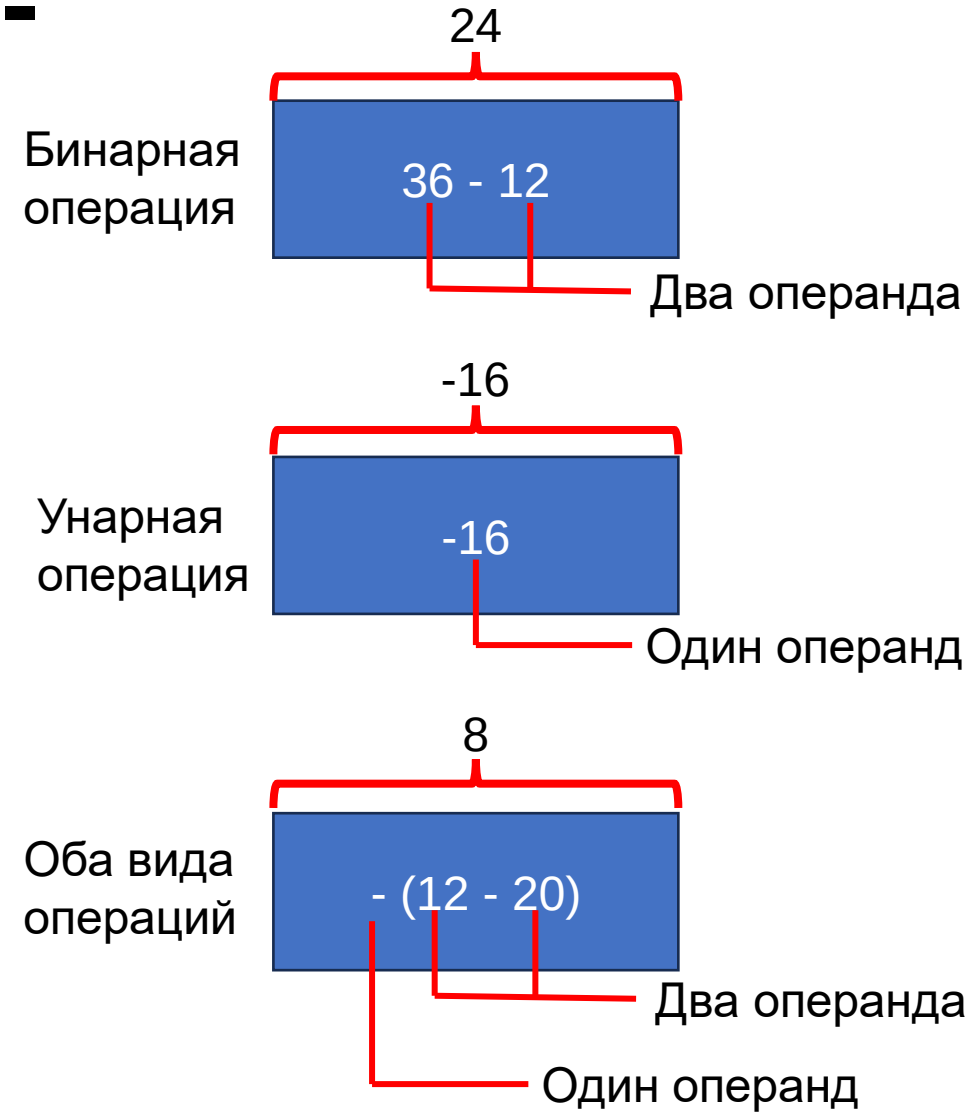
Примеры: printf("%d", 4 + 20); //24

income = salary + bribes;

- Операция вычитания вызывает вычитание числа, следующего за знаком -, из числа, находящегося перед этим знаком.

Примеры: takehome = 224.00 - 24.00;

rocky = -12; smokey = -rocky;



Операция умножения: *
Операция вычитания: -

- Умножение обозначается символом *.

Примеры: cm = 2.54 * inch;

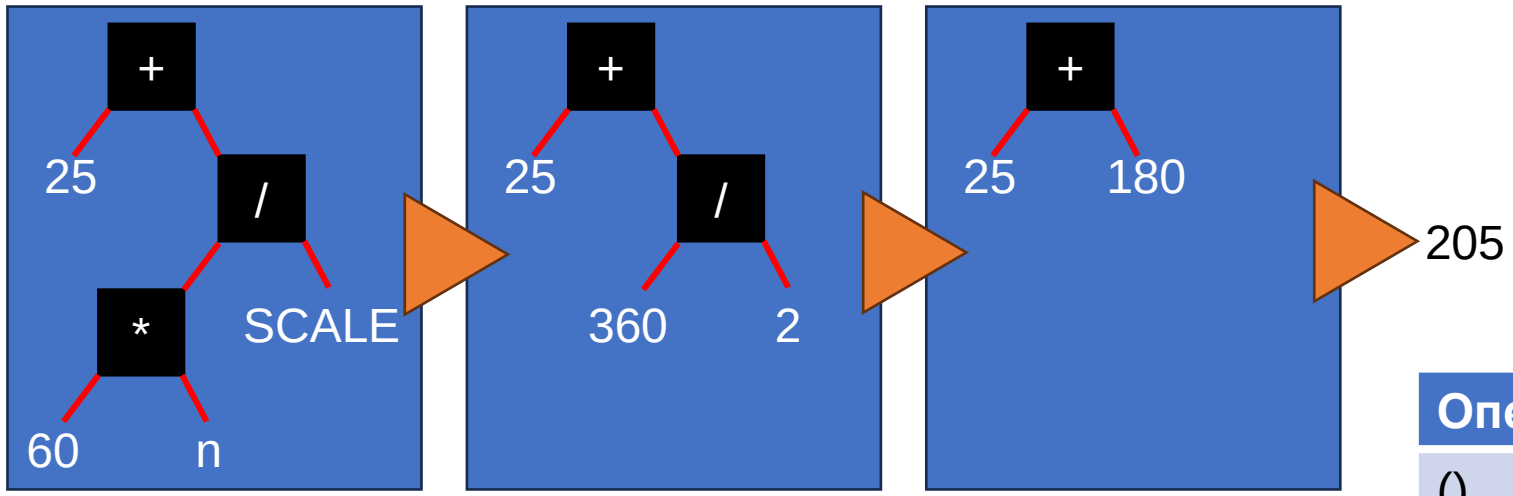
- В языке С символ / используется для обозначения деления.

```c
#include <stdio.h>
int main(void)
{
    printf("Целочисленное деление: 5/4 равно %d \n", 5 / 4);
    printf("Целочисленное деление: 6/3 равно %d \n", 6 / 3);
    printf("Целочисленное деление: 7/4 равно %d \n", 7 / 4);
    printf("Деление с плавающей запятой: 7./4. равно %1.2 f \n", 7. / 4.);
    printf("Смешанное деление: 7./4 равно % 1.2f \n", 7. / 4);
    return 0;
}
// Целочисленное деление: 5/4 равно 1
// Целочисленное деление: 6/3 равно 2
// Целочисленное деление: 7/4 равно 1
// Деление с плавающей запятой: 7./4. равно 1.75
// Смешанное деление: 7./4 равно  1.75
```

Приоритеты операций

a = 2 + 2 * 2;

```
SCALE = 2;
n = 6;
butter = 25 + 60.0 * n / SCALE
```



Операции	Ассоциативность
()	Слева направо
+ - (унарные)	Слева направо
* /	Слева направо
+ - (бинарные)	Слева направо
=	Слева направо

Операция sizeof и тип size_t

- С операция sizeof возвращает размер своего операнда в байтах. В языке С указано, что операция sizeof возвращает значение типа size_t. Это целочисленный тип без знака, но не совершенно новый тип.

```c
#include <stdio.h>
int main(void)
{
    system("chcp 65001");
    int n = 0;
    size_t intsize;
    intsize = sizeof(int);
    printf("n = %d, n состоит из %zd байтов; все значения int имеют %zd байтов.\n",
           n, sizeof n, intsize);
    return 0;
}
// n = 0, n состоит из 4 байтов; все значения int имеют 4 байтов
```

Операция деления по модулю: %

```c
#include <stdio.h>
#define SEC_PER_MIN 60 // количество секунд в минуте
int main(void)
{
    system("chcp 65001");
    unsigned int sec, min, left;
    printf("Перевод секунд в минуты и секунды!\n");
    printf("Введите количество секунд:\n");
    scanf("%u", &sec);          // чтение количества секунд
    min = sec / SEC_PER_MIN;  // усеченное количество минут
    left = sec % SEC_PER_MIN; // количество секунд в остатке
    printf("%u секунд - это %u минут(ы) %u секунд. ", sec, min, left);
    printf("Готово!\n");
    return 0;
}
// Перевод секунд в минуты и секунды!
// Введите количество секунд:
// 250
// 250 секунд - это 4 минут(ы) 10 секунд. Готово!
```

11 / 5 равно 2 и 11 % 5 равно 1
11 / -5 равно -2 и 11 % -2 равно 1
-11 / -5 равно 2 и -11 % -5 равно -1
-11 / 5 равно -2 и -11 % 5 равно -1

Операции инкремента и декремента: ++, --

```c
#include <stdio.h>
int main(void)
{
    int a = 1;
    int a1 = a++;
    int a2 = ++a;
    printf("a = %d, a1 = %d, a2 = %d\n", a, a1, a2);
    int b = 3;
    int b1 = b--;
    int b2 = --b;
    printf("b = %d, b1 = %d, b2 = %d\n", b, b1, b2);
    a = 1;
    b = 1;
    int n = (a + b++) * 6;
    printf("a = %d, b = %d, n = %d\n", a, b, n);
    return 0;
}
// a = 3, a1 = 1, a2 = 3
// b = 1, b1 = 3, b2 = 1
// a = 1, b = 2, n = 12
```

Операция инкремента решает простую задачу: она увеличивает (инкрементирует) значение своего операнда на 1. Существуют две разновидности этой операции. В первом случае символы ++ располагаются перед изменяемой переменной; это префиксная форма.

Префиксная форма

```c
q = 2*++a;
```

Сначала значение a увеличивается на 1, затем результат умножается на 2 и присваивается переменной q

Постфиксная форма

```c
q = 2*a++;
```

Сначала значение a умножается на 2 и присваивается переменной q, после чего значение a увеличивается на 1

Выражения

Выражение состоит из комбинации операций и операндов. Операнд — это то, над чем выполняется операция. Простейшим выражением является отдельный операнд, и он может служить отправной точкой для построения более сложных выражений.

Примеры выражений:

4

-6

4+21

a*(b+c/d)/20

q=5*2

x=++q%3

q>3

выражение	значение
-4 + 6	2
c = 3 + 8	11
5 > 3	1
6 + (c = 3 + 8)	17

Каждое выражение имеет значение!

Операторы

Операторы служат основными строительными блоками программы. Программа — это последовательность операторов с необходимыми знаками пунктуации. Оператор представляет собой завершенную инструкцию для компьютера. В языке С операторы распознаются по наличию точки с запятой в конце.

выражение

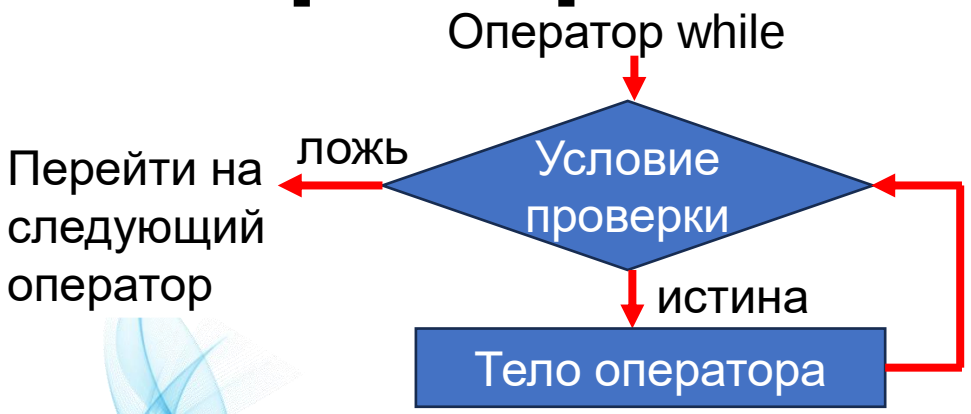
```
legs=4
```

оператор

```
legs=4;
```

```
; //пустой оператор

8;

3 + 4;

x = 25;

++ x;

y = sqrt (x);
```

Операторы



Составной оператор — это два или большее количество операторов, сгруппированных вместе путем помещения их в фигурные скобки;

```c
/* фрагмент 1 */
index = 0;
while (index++ < 10)
    sam = 10 * index + 2;
printf("sam= %d\n", sam);
```

```c
/* фрагмент 2 */
index = 0;
while (index++ < 10)
{
    sam = 10 * index + 2;
    printf("sam= %d\n", sam);
}
```

```c
#include <stdio.h>
int main(void)
{
    int count, sum;                 /* оператор объявления */
    count = 0;                      /* оператор присваивания */
    sum = 0;                        /* оператор присваивания */
    while (count++ < 20)            /* оператор */
        sum = sum + count;          /* while */
    printf("sum = %d\n", sum); /*оператор вызова функции */
    return 0;                       /*оператор возврата */
}
// sum = 210
```

Побочные эффекты и точки следования

Побочный эффект — это модификация объекта данных или файла. Например, побочный эффект оператора
states = 50;
Точка следования — это точка в ходе выполнения программы, в которой производится оценка всех побочных эффектов, прежде чем переходить на следующий шаг.
Полное выражение — это выражение, которое не является подвыражением более крупного выражения.
Пример 1:

```
while (guests++ < 10)
    printf("%d \n", guests);
```

Пример 2:

```
y = (4 + x++) + (6 + x++); // не стоит так писать
```

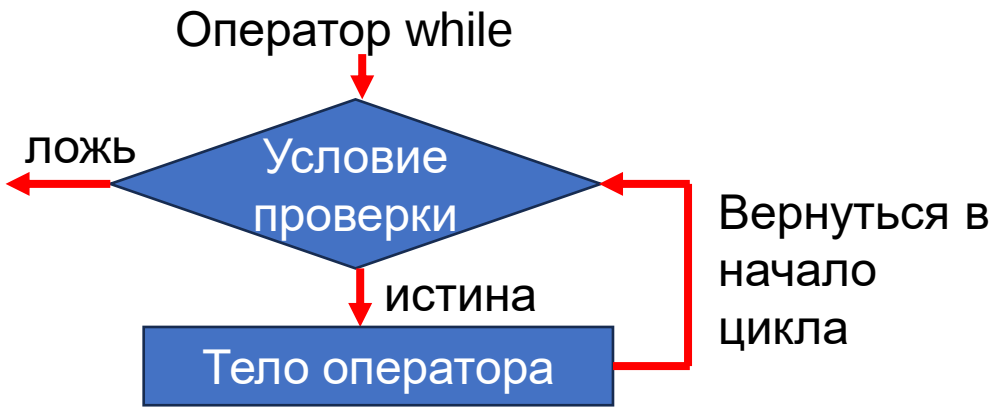
Адженда

Введение в данные в С	Ввод-вывод начальные сведения	Операции и операторы	Циклы и условия
15 минут	10 минут	25 минут	40 минут

Цикл while

```c
#include <stdio.h>
int main(void)
{
    system("chcp 65001");
    long num;
    long sum = 0L; /* инициализация переменной sum нулем */
    int status;
    printf("Введите целое число для последующего суммирования ");
    printf("(или q для завершения программы) : ");
    status = scanf("%ld", &num);
    while (status == 1) /* == обозначает равенство */
    {
        sum = sum + num;
        printf("Введите следующее целое число (или q для завершения программы): ");
        status = scanf("%ld", &num);
    }
    printf("Сумма введенных целых чисел равна %ld.\n", sum);
    return 0;
}
// Введите целое число для последующего суммирования (или q для завершения программы) : 44
// Введите следующее целое число (или q для завершения программы): 33
// Введите следующее целое число (или q для завершения программы): 88
// Введите следующее целое число (или q для завершения программы): 121
// Введите следующее целое число (или q для завершения программы): q
// Сумма введенных целых чисел равна 286.
```

Цикл чтения в стиле С




```
status = scanf("%ld", &num);
while (status == 1) {
    /* действия, выполняем в цикле */
    status = scanf("%ld", &num);
}
```

может быть заменена следующей:

```
while (scanf("%ld", &num) == 1)
{
    /* действия, выполняемое в цикле */
}
```

Завершение цикла while

Цикл while — это условный цикл, использующий входное условие (предусловие).

```c
#include <stdio.h>
int main(void)
{
    system("chcp 65001");
    int n = 5;
    while (n++ < 7)
    {
        printf("n = %d\n", n);
        n++;
        printf("Теперь n = %d\n", n);
        printf("Цикл завершен.\n");
    }
    return 0;
}
// n = 6
// Теперь n = 7
// Цикл завершен.
```

Особенности синтаксиса

Неправильное кодирование может приводить к бесконечному циклу

```c
#include <stdio.h>
int main(void)
{
    system("chcp 65001");
    int n = 0;
    while (n < 3)
        printf("n равно %d\n", n);
        n++;
    printf("Это все, что делает данная программа.\n");
    return 0;
}
// n равно 0
// n равно 0
// n равно 0
// n равно 0
// ...
```

Сравнение: операции и выражения отношений

Выражения с операциями сравнения называются ***выражениями отношений***, а операции, которые в них появляются — ***операциями отношений***.

Операция	Описание
<	Меньше
<=	Меньше или равно
==	Равно
>=	Больше или равно
>	Больше
!=	Не равно

Сравнение: операции и выражения отношений

```c
#include <math.h>
#include <stdio.h>
int main(void)
{
    system("chcp 65001");
    const double ANSWER = 3.14159;
    double response;
    printf("Каково значение числа pi?\n");
    scanf("%lf", &response);
    while (fabs(response - ANSWER) > 0.0001)
    {
        printf("Введите значение еще раз.\n");
        scanf("%lf", &response);
    }
    printf("Достаточно близко!\n");
    return 0;
}
// Каково значение числа pi?
// 3.14
// Введите значение еще раз.
// 3.1416
// Достаточно близко!
```

Что такое истина?

```c
#include <stdio.h>
int main(void)
{
    system("chcp 65001");
    int true_val, false_val;
    true_val = (10 > 2);   // значение истинного отношения
    false_val = (10 == 2); // значения ложного отношения
    printf("true = %d; false = %d \n", true_val, false_val);
    int n = 3;
    while (n)
        printf("%2d является истинным\n", n--);
    printf("%2d является ложным\n", n);
    n = -3;
    while (n)
        printf("%2d является истинным\n", n++);
    printf("%2d является ложным\n", n);
    return 0;
}
```

```
// true = 1; false = 0
//  3 является истинным
//  2 является истинным
//  1 является истинным
//  0 является ложным
// -3 является истинным
// -2 является истинным
// -1 является истинным
//  0 является ложным
```

Новый тип _Bool

В стандарте C99 для переменных, представляющих истинные и ложные значений был введен тип _Bool, который получил свое название в честь Джорджа Буля (George Boole), английского математика, который разработал алгебраическую систему, позволяющую формулировать и решать логические задачи.

```c
#include <stdio.h>
int main(void)
{
    system("chcp 65001");
    long num;
    long sum = 0L;
    _Bool input_is_good;
    printf("Введите целое число для последующего суммирования ");
    printf(" (или q для завершения программы) : ");
    input_is_good = (scanf("%ld", &num) == 1);
    while (input_is_good)
    {
        sum = sum + num;
        printf("Введите следующее целое число (или q для завершения программы): ");
        input_is_good = (scanf("%ld", &num) == 1);
    }
    printf("Сумма введенных целых чисел равна %ld.\n", sum);
    return 0;
}
```

Приоритеты операций

Операции (в порядке убывания приоритета)	Ассоциативность
( )	Слева направо
- + ++ -- sizeof (тип) (все унарные)	Слева направо
* / %	Слева направо
+ -	Слева направо
< > <= >=	Слева направо
== !=	Слева направо
=	Слева направо

Неопределенные циклы и циклы со счетчиком

Некоторые примеры цикла while представляли собой неопределенные циклы. Это означает, что заранее нельзя сказать, сколько раз цикл выполнится до того, как выражение станет ложным.

```c
#include <stdio.h>
int main(void)
{
    const int NUMBER = 5;
    int count = 1;           // инициализация
    while (count <= NUMBER) // проверка
    {
        printf("Будь моим другом! \n"); // действие
        count++;                        // обновление значения count}
    }
    return 0;
}
// Будь моим другом!
// Будь моим другом!
// Будь моим другом!
// Будь моим другом!
// Будь моим другом!
```

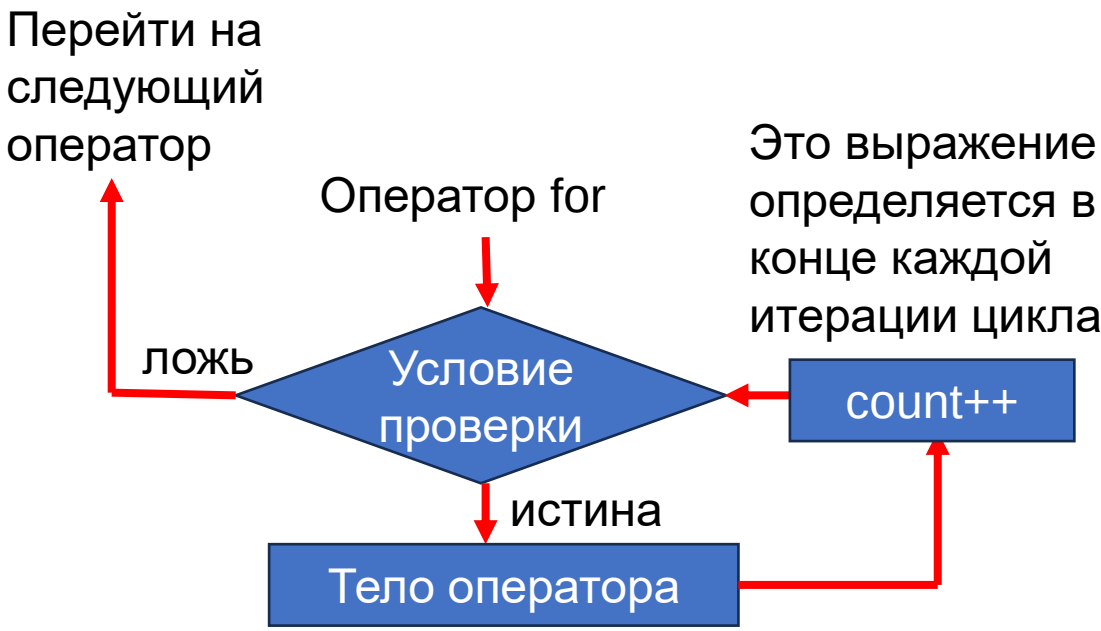
Цикл for

Некоторые примеры цикла while представляли собой неопределенные циклы. Это означает, что заранее нельзя сказать, сколько раз цикл выполнится до того, как выражение станет ложным.

```c
#include <stdio.h>
int main(void)
{
    const int NUMBER = 5;
    int count; // инициализация
    for (count = 0; count < NUMBER; count++)
        printf("Будь моим другом!\n");
    return 0;
}
// Будь моим другом!
// Будь моим другом!
// Будь моим другом!
// Будь моим другом!
// Будь моим другом!
```

Цикл for

```c
#include <stdio.h>
int main(void)
{
    int num;
    printf("    n   n в Кубе\n");
    for (num = 6; num >= 1; num--)
        printf("%5d %5d\n", num, num * num * num);
    return 0;
}
// n   n в Кубе
// 6   216
// 5   125
// 4    64
// 3    27
// 2     8
// 1     1
```



Дополнительные операции присваивания: +=, -=, *=, /=, %=

scores += 20 — то же, что и scores = scores + 20
dimes -= 2 — то же, что и dimes = dimes - 2
bunnies *= 2 — то же, что и bunnies = bunnies * 2
time /= 2.73 — то же, что и time = time / 2.73
reduce %= 3 — то же, что и reduce = reduce % 3
x *= 3 * y +12 — то же, что и x = x * (3* y +12)

рассмотренные операции присваивания имеют такой же низкий приоритет, как и операция =

Операция запятой

```c
#include <stdio.h>
int main(void)
{
    const int FIRST_OZ = 46; // тариф 2013 года
    const int NEXT_OZ = 20;  // тариф 2013 года
    int ounces, cost;
    printf("унции   тариф\n");
    for (ounces = 1, cost = FIRST_OZ; ounces <= 16; ounces++, cost += NEXT_OZ)
        printf("%5d $%4.2f\n", ounces, cost / 100.0);
    return 0;
}
// унции   тариф
//     1 $0.46
//     2 $0.66
//     3 $0.86
//    ...
//    16 $3.46
```

x = (y = 3, (z = ++y + 2) + 5); // 11 11 4

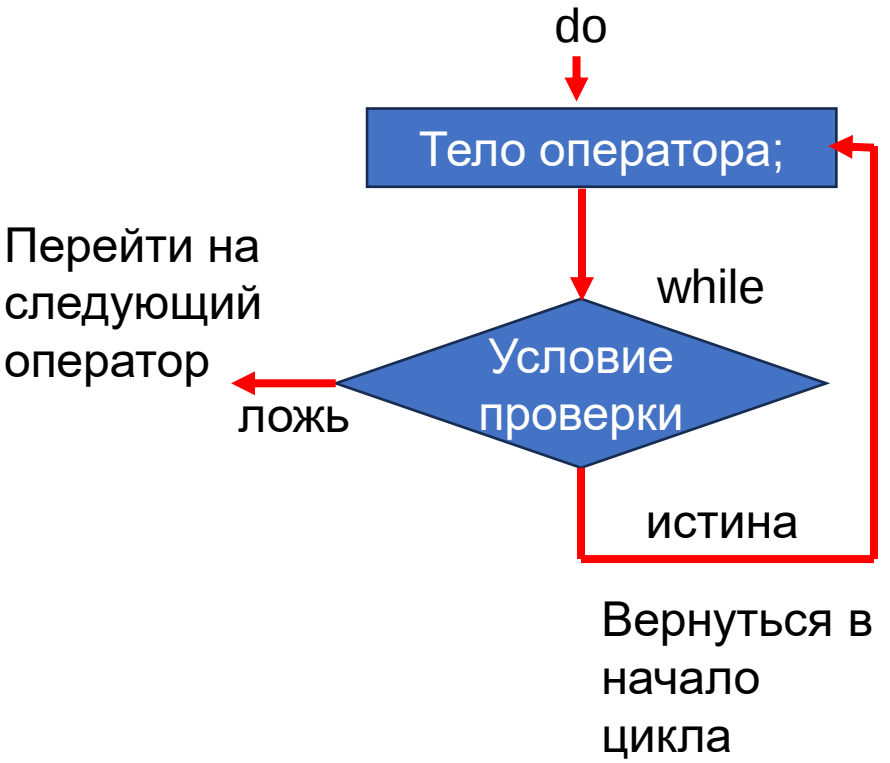
Греческий философ Зенон и цикл for

```c
#include <stdio.h>
int main(void)
{
    int t_ct; // счетчик элементов
    double time, power_of_2;
    int limit;
    printf("Введите желаемое количество элементов последовательности: ");
    scanf("%d", &limit);
    for (time = 0, power_of_2 = 1, t_ct = 1; t_ct <= limit; t_ct++, power_of_2 *= 2.0)
    {
        time += 1.0 / power_of_2;
        printf("время = %f, когда количество элементов = %d.\n", time, t_ct);
    }
    return 0;
}
// Введите желаемое количество элементов последовательности: 15
// время = 1.000000, когда количество элементов = 1.
// время = 1.500000, когда количество элементов = 2.
// время = 1.750000, когда количество элементов = 3.
// ...
// время = 1.999878, когда количество элементов = 14.
// время = 1.999939, когда количество элементов = 15.
```

Цикл с постусловием: do while

Цикл с постусловием – это цикл, в котором проверка условия производится после прохода каждой итерации цикла, благодаря чему гарантируется выполнение операторной части цикла минимум один раз.

```c
#include <stdio.h>
int main(void)
{
    const int secret_code = 13;
    int code_entered;
    do
    {
        printf("Чтобы войти в клуб лечения трискадекафобии, \n");
        printf("пожалуйста, введите секретный код: ");
        scanf("%d", &code_entered);
    } while (code_entered != secret_code);
    printf("Поздравляем! Вас вылечили!\n");
    return 0;
}
// Чтобы войти в клуб лечения трискадекафобии,
// пожалуйста, введите секретный код: 12
// Чтобы войти в клуб лечения трискадекафобии,
// пожалуйста, введите секретный код: 13
// Поздравляем! Вас вылечили!
```



Вложенные циклы

```c
#include <stdio.h>
#define ROWS 6
#define CHARS 10
int main(void)
{
    int row;
    char ch;
    for (row = 0; row < ROWS; row++)
    {
        for (ch = 'A'; ch < ('A' + CHARS); ch++)
            printf("%c", ch);
        printf("\n");
    }
    return 0;
}
// ABCDEFGHIJ
// ABCDEFGHIJ
// ABCDEFGHIJ
// ABCDEFGHIJ
// ABCDEFGHIJ
// ABCDEFGHIJ
```

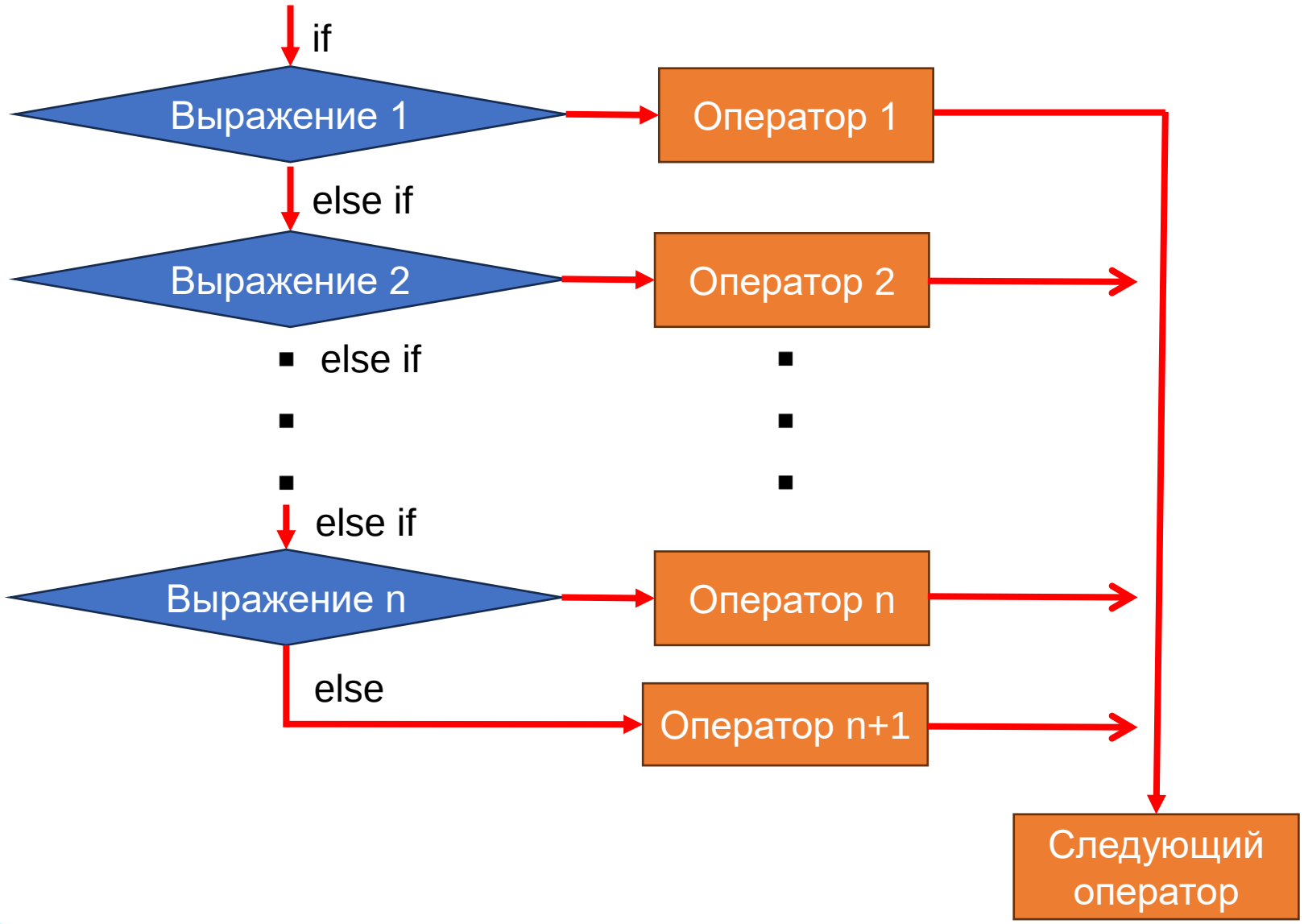
```c
#include <stdio.h>
int main(void)
{
    const int ROWS = 6;
    const int CHARS = 6;
    int row;
    char ch;
    for (row = 0; row < ROWS; row++)
    {
        for (ch = ('A' + row); ch < ('A' + CHARS); ch++)
            printf("%c", ch);
        printf("\n");
    }
    return 0;
}
// ABCDEF
// BCDEF
// CDEF
// DEF
// EF
// Fc
```

Оператор if

```c
#include <stdio.h>
int main(void)
{
    const int FREEZING = 0;
    float temperature;
    int cold_days = 0;
    int all_days = 0;
    printf("Введите список минимальных дневных температур.\n");
    printf("Используйте шкалу Цельсия; для завершения введите q.\n");
    while (scanf("%f", &temperature) == 1)
    {
        all_days++;
        if (temperature < FREEZING)
            cold_days++;
    }
    if (all_days != 0)
        printf("%d - общее количество дней: %.lf%% с температурой ниже нуля.\n",
               all_days, 100.0 * (float)cold_days / all_days);
    if (all_days == 0)
        printf("Данные не введены!\n");
    return 0;
}
// Введите список минимальных дневных температур.
// Используйте шкалу Цельсия; для завершения введите q.
// 12 5 -2.5 0 6 8 -3 -10 5 10 q
// 10 - общее количество дней: 30% с температурой ниже нуля.
```

Конструкция if else

```
if (выражение 1) {
    оператор 1
}
else if (выражение 2) {
    оператор 2
}
        .
        .
        .
else if (выражение n) {
    оператор n
}
else {
    оператор n+1
}
```



Задача решение 1\2

Дано:

Начисление за электричество идет по формуле

Первые 360 кВт/ч:

$0.13230 за 1 кВт/ч

Следующие 108 кВт/ч

$0.15040 за 1 кВт/ч

Следующие 252 кВт/ч

$0.30025 за 1 кВт/ч

Свыше 720 кВт/ч

$0.34025 за 1кВт/ч

Найти: сколько денег за **X** кв/ч

```c
#include <stdio.h>

#define RATE1 0.13230 // тариф за первые 360 кВт/ч
#define RATE2 0.15040 // тариф за следующие 108 кВт/ч
#define RATE3 0.30025 // тариф за следующие 252 кВт/ч
#define RATE4 0.34025 // тариф, когда расход принимает 720 кВт/ч
#define BREAK1 360.0  // первая точка разрыва тарифов
#define BREAK2 468.0  // вторая точка разрыва тарифов
#define BREAK3 720.0  // третья точка разрыва тарифов
#define BASE1 (RATE1 * BREAK1)
// стоимость 360 кВт/ч
#define BASE2 (BASE1 + (RATE2 * (BREAK2 - BREAK1)))
// стоимость 468 кВт/ч
#define BASE3 (BASE1 + BASE2 + (RATE3 * (BREAK3 - BREAK2)))
// стоимость 720 кВт/ч
```

Задача решение 2\2

```
int main(void)
{
    system("chcp 65001");
    double kwh;  // израсходованные киловатт-часы
    double bill; // сумма к оплате
    printf("Введите объем израсходованной электроэнергии в кВт/ч.\n");
    scanf("%lf", &kwh); // %lf для типа double
    if (kwh <= BREAK1)
        bill = RATE1 * kwh;
    else if (kwh <= BREAK2) // количество кВт/ч в промежутке от 360 до 468
        bill = BASE1 + (RATE2 * (kwh - BREAK1));
    else if (kwh <= BREAK3) // количество кВт/ч в промежутке от 468 до 720
        bill = BASE2 + (RATE3 * (kwh - BREAK2));
    else // количество кВт/ч превышает 720
        bill = BASE3 + (RATE4 * (kwh - BREAK3));

    printf("Сумма к оплате за %.1f кВт/ч составляет $%1.2f.\n", kwh, bill);
    return 0;
}
// Введите объем израсходованной электроэнергии в кВт/ч.
// 580
// Сумма к оплате за 580 кВт/ч составляет $97.50.
```

Операции и, или и не

```c
#include <stdio.h>
int main(void)
{
    system("chcp 65001");
    int a;
    printf("Введите целое число.\n");
    scanf("%d", &a);
    if (a % 2 == 0 && a % 3 == 0)
    {
        printf("%d делится на 6.\n", a);
    }
    if (a % 2 != 0 || a % 3 != 0)
    {
        printf("%d не делится на 6.\n", a);
    }
    if (!a)
    {
        printf("%d равно 0.\n", a);
    }
    return 0;
}
```

```
// Введите целое число.
// 6
// 6 делится на 6.s
// Введите целое число.
// 5
// 5 не делится на 6.
// Введите целое число.
// 0
// 0 делится на 6.
// 0 равно 0.
```

Альтернативное представление: iso646.h

```c
#include <stdio.h>
#include <iso646.h>
int main(void)
{
    system("chcp 65001");
    int a;
    printf("Введите целое число.\n");
    scanf("%d", &a);
    if (a % 2 == 0 and a % 3 == 0)
    {
        printf("%d делится на 6.\n", a);
    }
    if (a % 2 != 0 or a % 3 != 0)
    {
        printf("%d не делится на 6.\n", a);
    }
    if (not a)
    {
        printf("%d равно 0.\n", a);
    }
    return 0;
}
```

Традиционное представление	Представление посредством iso646.h	Описание
&&	and	И
||	or	Или
!	not	Не

```
// Введите целое число.
// 6
// 6 делится на 6.
// Введите целое число.
// 5
// 5 не делится на 6.
// Введите целое число.
// 0
// 0 делится на 6.
// 0 равно 0.
```

Приоритеты операций

Операции (в порядке убывания приоритета)	Ассоциативность
( )	Слева направо
! - + ++ -- sizeof (тип) (все унарные)	Слева направо
* / %	Слева направо
+ -	Слева направо
< > <= >=	Слева направо
== !=	Слева направо
&&	Слева направо
||	Слева направо
=	Слева направо

Условная операция ? :

Сокращенный способ представления одной из форм оператора **if else** — условное выражение, для которого применяется условная операция **? :**

Общая форма условного выражения:

Выражение1 ? выражение2 : выражение3

Если выражение1 имеет истинное (ненулевое) значение, то все условное выражение принимает то ж е значение, что и выражение 2. Если выражение имеет ложное (нулевое) значение, то вес условное выражение получает то же значение, что и выражение 3.

Пример: Условная операция ? :

```c
#include <stdio.h>
int main(void)
{
    system("chcp 65001");
    int num;
    printf("Введите число для проверки на четность:\n");
    while (scanf("%d", &num) == 1)
    {
        printf("Число %d является %s.\n", num,
               num % 2 == 0 ? "четным" : "не четным");
        printf("Введите следующее значение (или q для завершения):\n");
    }
    return 0;
}
// Введите число для проверки на четность:
// 64
// Число 64 является четным.
// Введите следующее значение (или q для завершения):
// 65
// Число 65 является не четным.
// Введите следующее значение (или q для завершения):
// q
```

Оператор continue

- Когда он встречается, он вызывает пропуск оставшейся части итерации и начало новой итерации.
- Если оператор continue указан внутри вложенной структуры, он воздействует только на самую внутреннюю структуру, содержащую его.
- Не применяйте оператор continue, если он вместо упрощения усложняет код.

```c
#include <stdio.h>
int main(void)
{
    system("chcp 65001");
    int sum = 0;
    for (int i = 0; i < 10; i++)
    {
        if (i % 2 == 0)
            continue;
        sum += i;
    }
    printf("Сумма нечетных от 0 до 10 равна %d", sum);
    return 0;
}
// Сумма нечетных от 0 до 10 равна 25
```

Оператор break

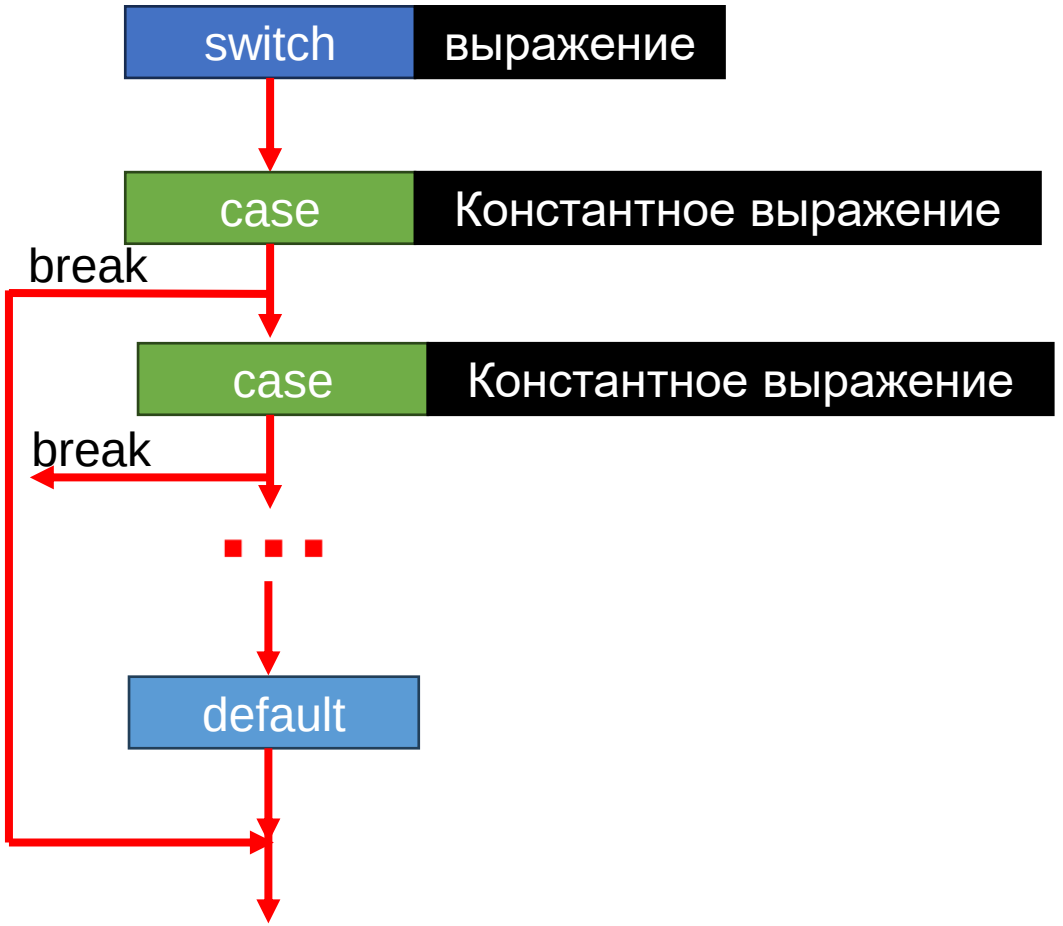
- Оператор break в цикле заставляет программу прервать цикл и перейти к выполнению следующего оператора. Если оператор break находится внутри вложенных циклов, его действие распространяется только на самый внутренний цикл, в котором он содержится..

```c
#include <stdio.h>
int main(void)
{
    float length, width;
    printf("Введите длину прямоугольника:\n");
    while (scanf("%f", &length) == 1)
    {
        printf("Длина = %0.2f:\n", length);
        printf("Введите ширину прямоугольника: \n");
        if (scanf("%f", &width) != 1)
            break;
        printf("Длина = %0.2f:\n", width);
        printf("Площадь = %0.2f:\n", length * width);
        printf("Введите длину прямоугольника:\n");
    }
    printf("Программа завершена.\n");
    return 0;
}
```

```
// Введите длину прямоугольника:
// 34
// Длина = 34.00:
// Введите ширину прямоугольника:
// 56
// Длина = 56.00:
// Площадь = 1904.00:
// Введите длину прямоугольника:
// o
// Программа завершена.
```

Использование оператора switch

```
switch (expression)
{
case /* constant-expression */:
    /* code */
    break;

default:
    break;
}
```



Использование оператора switch

```
_Bool input_is_good = 1;
int num_1, num_2;
char symbol;
printf("Enter expression.\n");
while (input_is_good)
{
    input_is_good = scanf("%d %c %d", &num_1, &symbol, &num_2);
    switch (symbol)
    {
    case '+':
        printf("%d\n", num_1 + num_2);
        break;
    case '-':
        printf("%d\n", num_1 - num_2);
        break;
    case '*':
        printf("%d\n", num_1 * num_2);
        break;
    case '/':
        printf("%d\n", num_1 / num_2);
        break;
    default:
        input_is_good = 0;
        printf("Exit");
        break;
    }
}
```

```
// Enter expression.
// 1 + 5
// 6
// 3 - 2
// 1
// 4 * 3
// 12
// 15 / 3
// 5
// 16 % 3
// Exit
```

Оператор goto

Избегайте goto. Чрезмерно используя goto, вы создаете лабиринт в потоке управления программы.

```c
#include <stdio.h>
int main(void)
{
    int a, b, c;
    scanf("%d %d %d", &a, &b, &c);
    if (a > 14)
        goto one;
    b = 2;
    goto two;
one:
    a = 3;
two:
    b = 2 * c;
    printf("a = %d b = %d c = %d\n", a, b, c);
    return 0;
}
// 15 25 35
// a = 3 b = 70 c = 35
// 10 20 30
// a = 10 b = 60 c = 30
```

Пример где может пригодится

```
while (funct > 0) {
    for (i = 1, i <= 100; i++) {
        for (j = 1; j <= 50; j + + ) {
            множество операторов;
            if (признак ошибки)
                goto help;
            операторы;
        }
        еще множество операторов;
    }
    другое множество операторов;
}
третье множество операторов;
help : код устранения ошибки;
```

Резюме

- Мы познакомились с типами данных в С и разобрали подробно целочисленные типы
- Вкратце поговорили о вводе выводе на консоль и в файл
- Мы разобрали множество операций и посмотрели их приоритеты
- Мы рассмотрели циклы for, while, do while
- Мы рассмотрели условия if else, switch, goto