Синтаксис

do *statement* while (*expression*) ;

Выражение ***expression*** в операторе **do-while** вычисляется после выполнения тела цикла. Поэтому тело цикла всегда выполняется по крайней мере один раз.

Выражение ***expression*** должно иметь арифметический тип или тип указателя. Выполнение происходит следующим образом:

1 Выполняется тело оператора.

2 Затем вычисляется значение ***expression***. Если выражение ***expression*** имеет значение **false**, выполнение оператора **do-while** завершается и управление передается следующему оператору программы. Если ***expression*** имеет значение **true** (то есть не равно нулю), процесс повторяется с шага 1.

Выполнение оператора **do-while** также прерывается, когда в теле оператора выполняется оператор **break**, **goto** или **return** .

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>

enum MENU {start = 1, quit};

int main(void) {
  setlocale(LC_ALL, "RU");

  puts("Приветствие\n");

  puts("1 - Выполнить программу");
  puts("2 - Завершить работу");

  int userChoice = 0;

  do {
    printf("\nВыберите пункт меню: ");
    scanf("%d", &userChoice);

    switch (userChoice) {
      case start:
        puts("\nВыполнение расчета");
        break;
      case quit:
        break;
      default:
        puts("\n1 - Выполнить программу");
        puts("2 - Завершить работу");
        break;
    }
  } while (userChoice != quit);

  return EXIT_SUCCESS;
}
```

Синтаксис

`for ( `*`init-expression`*$_{\text{необ.}}$` ; `*`cond-expression`*$_{\text{необ.}}$` ; `*`loop-expression`*$_{\text{необ.}}$` )` ***`statement`***

Оператор **`for`** выполняется следующим образом:

1 Вычисляется выражение ***`init-expression`*** (если есть). Оно определяет инициализацию цикла. На тип выражения ***`init-expression`*** ограничений не накладывается.

2 Вычисляется выражение ***`cond-expression`*** (если есть). Это выражение должно иметь арифметический тип или тип указателя. Оно вычисляется перед каждой итерацией. Возможны три результата.

- Если выражение ***`cond-expression`*** возвращает **`true`** (ненулевое значение), выполняется оператор ***`statement`***, после чего вычисляется выражение ***`loop-expression`*** (при его наличии). Выражение ***`loop-expression`*** вычисляется после завершения каждой итерации. На его тип ограничений не накладывается. Затем процесс начинается снова с вычисления выражения ***`cond-expression`***.

Синтаксис

`for` **`(`** ***`init-expression`***$_{\text{необ.}}$ **`;`** ***`cond-expression`***$_{\text{необ.}}$ **`;`** ***`loop-expression`***$_{\text{необ.}}$ **`)`** ***`statement`***

- Если выражение ***`cond-expression`*** опущено, оно (***`cond-expression`***) считается равным **`true`**, а выполнение продолжается согласно описанию в предыдущем абзаце. Выполнение оператора **`for`** без аргумента ***`cond-expression`*** завершается только при выполнении в его теле оператора **`break`** или **`return`** либо при выполнении оператора перехода **`goto`** к оператору с меткой вне тела оператора **`for`** .

- Если выражение ***`cond-expression`*** имеет значение **`false`** (0), выполнение оператора **`for`** завершается и управление передается следующему оператору программы.

Выполнение оператора **`for`** также завершается при выполнении в его теле оператора **`break`** , **`goto`** или **`return`** . Оператор **`continue`** в цикле **`for`** задает дальнейшее вычисление выражения ***`loop-expression`***. При выполнении оператора **`break`** в цикле for выражение ***`loop-expression`*** не вычисляется и не выполняется.

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {

  for (int i = 10; i > 0; i--) {
    printf("%d\n", i);
  }

  return EXIT_SUCCESS;
}
```

```c
 for (int i = 10; i > 0; printf("%d\n", i--))
    ;
```

Базовые конструкции

От ***0*** до ***n - 1***

```
for (i = 0; i < n; i++)
```

От ***1*** до ***n***

```
for (i = 1; i <= n; i++)
```

От ***n - 1*** до ***0***

```
for (i = n - 1; i >= 0; i--)
```

От ***n*** до ***1***

```
for (i = n; i > 0; i--)
```

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {

  int i = 10;

  for (; i > 0; i--) {
    printf("%d\n", i);
  }

  return EXIT_SUCCESS;
}
```

```c
  for (; i > 0;) {
    printf("%d\n", i--);
  }
```

```c
  for (int i = 10; i > 0;) {
    printf("%d\n", i--);
  }

  printf("%d\n", i);
```

```c
  int i, sum, n = 10;

  for (i = 1, sum = 0; i <= n; i++) {
    printf("%d\n", i);
    sum +=i;
  }

  printf("%d\n", sum);
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <stdbool.h>

int main(void) {
  setlocale(LC_ALL, "RU");

  int number = 0;
  bool isPrime = true;

  scanf("%d", &number);

  for (int i = 2; i <= number / 2; i++) {
    if (number % i == 0) {
      isPrime = false;
      break;
    }
  }

  if (isPrime) {
    puts("Число является простым");
  }
  else {
    puts("Число не является простым");
  }

  return EXIT_SUCCESS;
}
```

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
  int number = 0, count = 0, sum = 0;

  const int total = 5;

  while (count < total) {
    scanf("%d", &number);

    if (number == 0) {
      continue;
    }

    sum += number;
    count++;
  }
  return EXIT_SUCCESS;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>

enum MENU {start = 1, quit};

int main(void) {
  setlocale(LC_ALL, "RU");

  puts("Приветствие\n");

  puts("1 - Выполнить программу");
  puts("2 - Завершить работу");

  int userChoice = 0;

  for(;;) {
    printf("\nВыберите пункт меню: ");
    scanf("%d", &userChoice);

    switch (userChoice) {
      case start:
        puts("\nВыполнение расчета");
        break;
      case quit:
        return EXIT_SUCCESS;
      default:
        puts("\n1 - Выполнить программу");
        puts("2 - Завершить работу");
        break;
    }
  }
  return EXIT_SUCCESS;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>

double GetDouble(void) {
  double input = 0.0;

  while (!scanf("%lf", &input)) {
    while (getchar() != '\n')
      ;
    printf("Ошибка ввода. Введите вещественное число.\n");
  }

  while (getchar() != '\n')
    ;

  return input;
}

int main(void) {
  setlocale(LC_ALL, "RU");

  double x = 0.0, y = 0.0;

  puts("Введите вещественное значение переменной x");
  x = GetDouble();

  puts("Введите вещественное значение переменной y");
  y = GetDouble();

  printf("x = %f\ny = %f\n", x, y);

  return EXIT_SUCCESS;
}
```