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

#define FREEZING_POINT 32.0
#define SCALE_FACTOR (5.0 / 9.0)

int main(void) {
  setlocale(LC_ALL, "RU");
  double fahrenheit = 0.0;

  printf("Температура по шкале Фаренгейта: ");
  scanf("%lf", &fahrenheit);

  double celsius = (fahrenheit - FREEZING_POINT) * SCALE_FACTOR;

  printf("Температура по шкале Цельсия = %g\n", celsius);
  return EXIT_SUCCESS;
}
```

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

#define FREEZING_POINT 32.0
#define SCALE_FACTOR (5.0 / 9.0)

int main(void) {
  setlocale(LC_ALL, "RU");
  double fahrenheit = 0.0;

  printf("Температура по шкале Фаренгейта: ");
  scanf("%lf", &fahrenheit);

  const double normalHumanBodyTemperature = 98.6;

  if (fahrenheit == normalHumanBodyTemperature) {
    puts("\nСовпадает с 98,6 °F\n");
  }

  double celsius = (fahrenheit - FREEZING_POINT) * SCALE_FACTOR;

  printf("Температура по шкале Цельсия = %g\n", celsius);
  return EXIT_SUCCESS;
}
```

Синтаксис

`if ( expression ) statement`

`if ( expression ) statement else statement`

В обоих формах оператора **`if`** выражение может иметь любое значение, кроме структуры, и его вычисление влечет за собой все соответствующие побочные эффекты.

В первой форме синтаксиса оператор ***`statement`*** выполняется, если значение ***`expression`*** имеет значение **`true`** (не равно нулю). Если ***`выражение`*** возвращает **`false`**, ***`statement`*** пропускается.

Во второй форме синтаксиса с **`else`** , если ***`expression`*** имеет значение **`false`**, выполняется второй оператор ***`statement`***.

После этого управление передается (в обеих формах) из оператора **`if`** в следующий по порядку оператор программы, если выполняемый оператор не содержит операторов **`break`** , **`continue`** или **`goto`** .

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

#define FREEZING_POINT 32.0
#define SCALE_FACTOR (5.0 / 9.0)

int main(void) {
  setlocale(LC_ALL, "RU");
  double fahrenheit = 0.0;

  printf("Температура по шкале Фаренгейта: ");
  scanf("%lf", &fahrenheit);

  const double normalHumanBodyTemperature = 98.6;

  if (fahrenheit == normalHumanBodyTemperature) {
    puts("\nСовпадает с 98,6 °F\n");
  }
  else {
    puts("\nОтличается от 98,6 °F\n");
  }

  double celsius = (fahrenheit - FREEZING_POINT) * SCALE_FACTOR;

  printf("Температура по шкале Цельсия = %g\n", celsius);
  return EXIT_SUCCESS;
}
```

```c
  if (fahrenheit != normalHumanBodyTemperature) {
    puts("\nОтличается от 98,6 °F\n");
  }
```

```c
  fahrenheit == normalHumanBodyTemperature ?
    puts("\nСовпадает с 98,6 °F\n") : puts("\nОтличается от 98,6 °F\n");
```

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

#define FREEZING_POINT 32.0
#define SCALE_FACTOR (5.0 / 9.0)

int main(void) {
  setlocale(LC_ALL, "RU");
  double fahrenheit = 0.0;

  printf("Температура по шкале Фаренгейта: ");
  scanf("%lf", &fahrenheit);

  const double normalHumanBodyTemperature = 98.6;
  const double lowerNormalBound = 97.0;
  const double upperNormalBound = 99.0;

  if (fahrenheit == normalHumanBodyTemperature) {
    puts("\nСовпадает с 98,6 °F");
  }
  else {
    puts("\nОтличается от 98,6 °F");
  }

  if (fahrenheit >= lowerNormalBound && fahrenheit <= upperNormalBound) {
    puts("\nВ пределах нормы");
  }

  double celsius = (fahrenheit - FREEZING_POINT) * SCALE_FACTOR;

  printf("\nТемпература по шкале Цельсия = %g\n", celsius);
  return EXIT_SUCCESS;
}
```

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

#define FREEZING_POINT 32.0
#define SCALE_FACTOR (5.0 / 9.0)

int main(void) {
  setlocale(LC_ALL, "RU");
  double fahrenheit = 0.0;

  printf("Температура по шкале Фаренгейта: ");
  scanf("%lf", &fahrenheit);

  const double normalHumanBodyTemperature = 98.6;
  const double lowerNormalBound = 97.0;
  const double upperNormalBound = 99.0;

  if (fahrenheit == normalHumanBodyTemperature) {
    puts("Совпадает с 98,6 °F");
  }
  if (fahrenheit != normalHumanBodyTemperature) {
    puts("Отличается от 98,6 °F");
  }

  if (fahrenheit >= lowerNormalBound && fahrenheit <= upperNormalBound) {
    puts("\nВ пределах нормы");
  }
  else {
    if (fahrenheit < lowerNormalBound) {
      puts("\nХолоднее");
    }
    else {
      puts("\nГорячее");
    }
  }

  double celsius = (fahrenheit - FREEZING_POINT) * SCALE_FACTOR;

  printf("Температура по шкале Цельсия = %g\n", celsius);
  return EXIT_SUCCESS;
}
```

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

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

  int grade = 0;
  scanf("%d", &grade);

  if (grade == 5) {
    puts("\nОтлично");
  }
  else if (grade == 4) {
    puts("\nХорошо");
  }
  else if (grade == 3) {
    puts("\nУдовлетворительно");
  }
  else if (grade == 2) {
    puts("\nНеудовлетворительно");
  }
  else if (grade == 1) {
    puts("\nПлохо");
  }
  else if (grade == 0) {
    puts("\nХуже некуда");
  }
  else {
    puts("\nНекорректная оценка");
  }

  return EXIT_SUCCESS;
}
```

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

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

  int grade = 0;
  scanf("%d", &grade);

  switch (grade) {
    case 5:
      puts("\nОтлично");
      break;
    case 4:
      puts("\nХорошо");
      break;
    case 3:
      puts("\nУдовлетворительно");
      break;
    case 2:
      puts("\nНеудовлетворительно");
      break;
    case 1:
      puts("\nПлохо");
      break;
    case 0:
      puts("\nХуже некуда");
      break;
    default:
      puts("\nНекорректная оценка");
      break;
  }

  return EXIT_SUCCESS;
}
```

Синтаксис

switch (*expression*) *statement*

***labeled-statement*:**
case *constant-expression* : *statement*
default : *statement*

Оператор **switch** передает управление одному из ***labeled-statement*** в своем теле в зависимости от значения ***expression*** .

Значения ***expression*** и значение каждого ***constant-expression*** должны иметь целочисленный тип. Выражение ***constant-expression*** должно иметь однозначное константное целочисленное значение во время компиляции.

Управление передается оператору **case** , значение ***constant-expression*** которого соответствует значению выражения ***expression*** . Оператор **switch** может содержать неограниченное число экземпляров **case** . Однако значения ни одной из пар выражений ***constant-expression*** в одном операторе **switch** не должны совпадать. Выполнение тела оператора **switch** начинается с первого соответствующего оператора ***labeled-statement*** или после него. Выполнение продолжается до конца тела оператора или пока оператор **break** не передаст управление за пределы тела.

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

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

  int grade = 0;
  scanf("%d", &grade);

  switch (grade) {
    case 5:
      puts("\nОтлично");
    case 4:
      puts("\nХорошо");
    case 3:
      puts("\nУдовлетворительно");
    case 2:
      puts("\nНеудовлетворительно");
    case 1:
      puts("\nПлохо");
    case 0:
      puts("\nХуже некуда");
    default:
      puts("\nНекорректная оценка");
  }

  return EXIT_SUCCESS;
}
```

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

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

  int grade = 0;
  scanf("%d", &grade);

  switch (grade) {
    case 5:
    case 4:
    case 3:
      puts("\nЗачет");
      break;
    case 2:
    case 1:
    case 0:
      puts("\nНезачет");
      break;
    default:
      puts("\nНекорректная оценка");
      break;
  }

  return EXIT_SUCCESS;
}
```

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

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

  int grade = 0;
  scanf("%d", &grade);

  int numberOfPassed = 0;
  int total = 0;

  switch (grade) {
    case 5: case 4: case 3:
      numberOfPassed++;
      // Продолжение выполнения
    case 2: case 1: case 0:
      total++;
      break;
    default:
      puts("\nНекорректная оценка");
      break;
  }

  return EXIT_SUCCESS;
}
```

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

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

  puts("Приветствие\n");

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

  int userChoice = 0;
  scanf("%d", &userChoice);

  switch (userChoice) {
    case 1:
      puts("\nВыполнение расчета");
      break;
    case 2:
      break;
    default:
      puts("\nПункт меню не существует");
      break;
  }

  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;
  scanf("%d", &userChoice);

  switch (userChoice) {
    case start:
      puts("\nВыполнение расчета");
      break;
    case quit:
      break;
    default:
      puts("\nПункт меню не существует");
      break;
  }

  return EXIT_SUCCESS;
}
```

Синтаксис

`while ( `*`expression`*` ) `*`statement`

Выражение ***`expression`*** должно иметь арифметический тип или тип указателя.

Выполнение происходит следующим образом:

1 Вычисляется значение ***`expression`***.

2 Если выражение ***`expression`*** изначально ложно, тело оператора **`while`** не выполняется ни одного раза, а управление передается от оператора **`while`** следующему оператору в программе.

Если ***`expression`*** имеет значение **`true`** (то есть не равно нулю), выполняется тело оператора и процесс повторяется с шага 1.

Выполнение оператора **`while`** прерывается, если в теле оператора выполняется оператор **`break`** , **`goto`** или **`return`** . Для прерывания итерации без выхода из цикла **`while`** используется оператор **`continue`** . Оператор **`continue`** передает управление в следующую итерацию оператора **`while`** .

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

int main(void) {

  unsigned int counter = 1;

  while (counter <= 10) {
    printf("%u\n", counter);
    counter++;
  }

  return EXIT_SUCCESS;
}
```

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

int main(void) {

  unsigned int counter = 0;

  while (++counter <= 10) {
    printf("%u\n", counter);
  }

  return EXIT_SUCCESS;
}
```