5555

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. В.Ф. УТКИНА

А.А. Большакова, С.С. Лукша, Е.В. Тишковец

Методические указания к практическим занятиям по курсу *«Основы программирования на языке С»*

Рязань 2020

УДК 004.42

Основы программирования на языке С: методические указания / А.А. Большакова, С.С. Лукша, Е.В. Тишковец; Рязан. гос. радиотехн. ун-т. им. В.Ф. Уткина – Рязань, 2020. – 32 с.

Рассмотрены ключевые вопросы программирования на языке С мобильных роботов на платформе LEGO Mindstorm NXT с использованием среды RobotC. Представлены функциональные возможности по взаимодействии с аппаратным обеспечением. Описан синтаксис ключевых функций и операторов.

Методические указания предназначены для школьников старше 10 лет, которые проходят обучения по курсам дополнительного образования, связанными с робототехникой.

Робототехника, программирование, язык С, мобильный робот.

Печатается по решению редакционно-издательского совета Рязанского государственного радиотехнического университета.

Рецензент: кафедра информационно-измерительной и биомедицинской техники Рязанского государственного радиотехнического университета.

Большакова Анастасия Андреевна
Лукша Сергей Сергеевич
Тишковец Елизавета Витальевна

Основы программирования на языке С

Заказ
Рязанский государственный радиотехнический университет
им. В.Ф. Уткина.
390005, Рязань, ул. Гагарина, 59/1. Редакционно-издательский центр РГРТУ.

Введение

В современном мире программирование затрагивает практически любую область науки и техники. Даже те, кто не являются программистами, должны уметь формализовать задачи реального мира так, чтобы их могла выполнить машина. Именно поэтому важно знать ключевые аспекты программирования, и в этом курсе они будут рассмотрены с использованием языка С.

Язык программирования – формальный язык, предназначенный для записи компьютерных программ. Язык программирования определяет набор лексических, синтаксических и семантических правил, формирующих внешний вид программы и действия, которые выполнит исполнитель (как правило, ЭВМ) под её управлением. Языки программирования делятся по уровню:

– языки низкого уровня – содержат ограниченный набор инструкций и предназначены для создания списка команд непосредственно для ЭВМ, самый низкий уровень имеет машинный код;

– языки высокого уровня, которые предназначены для удобства использования программистом.

Можно также разделить языки программирования по способу реализации:

– *компилируемый* – язык программирования, исходный код которого преобразуется компилятором в машинный код и записывается в файл с особым заголовком и/или расширением для последующей идентификации этого файла, как исполняемого операционной системой;

– *интерпретируемый* – язык программирования, исходный код на котором выполняется методом интерпретации, то есть последовательного перевода команд на машинный код.

Исходный код (исходный текст) – текст компьютерной программы на каком-либо языке программирования или языке разметки, который может быть прочтён человеком. Исходный код транслируется в исполняемый код целиком до запуска программы при помощи компилятора или может исполняться сразу при помощи интерпретатора.

Интегрированная среда разработки, ИСР (англ. Integrated development environment — IDE) – комплекс программных средств, используемый программистами для разработки программного обеспечения (ПО). Среда разработки включает в себя:

– текстовый редактор – программный инструмент для набора и редактирования исходного кода;

– транслятор (компилятор и/или интерпретатор) – программный инструмент для перевода исходного кода в машинный язык;

– средства автоматизации сборки – обеспечивают поиск и подключение программных библиотек, необходимых для работы ПО;

– отладчик – программный инструмент для автоматизации процесса поиска ошибок. В зависимости от встроенных возможностей, отладчик позволяет выполнять трассировку, отслеживать, устанавливать или изменять значения переменных в процессе выполнения кода, устанавливать и удалять контрольные точки или условия остановки и так далее.

В ходе изучения принципов программирования нами будет использована среда разработки *RobotC*. С её помощью будут создаваться программы для мобильных роботов, созданных с использование конструкторов LEGO Mindstorm NXT.

1. Комментирование кода программы

При написании кода программы очень важно уделять внимание комментариям, поскольку они помогают быстрее разобраться в ранее написанном коде, а также понять его человеку, впервые увидевшему данную программу. В языке С для написания комментариев в коде программы имеются специальные методы.

Первый способ – это пара наклонных линий (слэш, или знак деления) – //. Если в тексте программы встречается такой символ, то всё написанное от него и до конца строки считается комментарием и при компилировании не учитывается

```
//Объявление переменных
int x
//Присвоение переменной значения
x = 5;
```

В данном коде используются две операции, перед каждой из которых идут комментарии (выделены зелёным), которые сообщают то, что делается в следующей строчке. Возможно написание комментария на продолжении строчки, например:

```
int y       //Объявление и инициализация переменной
```

Второй способ комментирования применяется в том случае, когда комментарий занимает более одной строки. Для обозначения начала комментария используется символ /*, для обозначения конца комментария */.

```
/* В данной программе будут изучены
основы работы с переменными
в языке С */
int x = 0;
```

Комментарий выделен зелёным цветом. Его длина не ограничена, но обязательное условие – текст комментария должен располагаться между символами начала /* и конца */.

2. Поле редактирования кода

При написании программы всегда используется главная функция программы, в которой осуществляется написание основного кода. Эта функция является первой, к которой обращается контроллер при запуске программы. В среде ROBOTC такой функцией является task main. Все действия, которые будут выполняться в программе, должны быть описаны в этой функции, а точнее в её теле, то есть между двумя фигурными скобками.

```
task main

//Здесь располагается текст программы

}
```

При использовании глобальных переменных или дополнительных функций допускается написание кода за пределами task main(). Эти случаи будут рассмотрены позднее.

3. Вывод информации на экран контроллера

Одним из основных шагов при работе с контроллером робототехнической платформы является вывод информации на экран. Для выполнения подобных действий в языке С используются функции. По внешнему виду функции похожи на те, что используются в обычной математике. Например в тригонометрии используются функции sin x и cos x. При этом sin – это название функции, а x – это её аргумент. В языке С принято аргумент писать в круглых скобках, например sin(x). Кроме этого, количество аргументов может быть больше одного и в этом случае они перечисляются через запятую. Для вывода на экран текстовой информации используется функция:

```
nxtDisplayTextLine(1, "Hello world");
```

Сначала пишется название функции, то есть nxtDisplayTextLine, затем идут круглые скобки, в которых указываются аргументы функции. В данном случае их два. Первый аргумент, (цифра 1 в примере) – это номер строки, на которую будет выводиться текст. Второй аргумент (“Hello word” в примере) – это сам текст, который будет выведен на экране контроллера. Текст пишется в кавычках.

Кроме этого, помимо функции вывода на экран имеется функция задержки, служащая для того, чтобы программа «заснула», а мы за это время

успели прочитать сообщение с экрана. Период задежки может быть задан в качестве аргумента функции. Пример показан ниже

```
wait1Msec(10000);
```

Функция выполняет задержку на 1 миллисекунду, а аргумент в скобках обозначает количество повторений такой задержки. Поскольку в одной секунде 1000 миллисекунд, то для реализации задержки на 10 секунд необходимо указать значение аргумента 10 000.

Задание 3.1

Используя функцию nxtDisplayTextLine, выведите на экран контроллера свои фамилию, имя и отчество так, чтобы они располагались последовательно на каждой новой строчке.

4. Переменные в языке С

Одним из основных элементов языка С являются переменные. В математике мы часто используем различные функции, в которых можно встретить переменные. Например, в функции

$$y = 3x + 6$$

x и y – это переменные, причём значение y зависит от значения x. При программировании мы также можем использовать переменные, но здесь есть особенность: прежде, чем использовать переменную, нужно определить её тип. Мы привыкли иметь дело с целыми числами, причём как с положительными, так и с отрицательными, такими как, например, 1, 2, -5, -156, и т.д. и с дробными, такими как 1.5, 2.96, -9.58 и т.д. Поэтому, в языке С можно выделить два основных типа числовых переменных: это целый тип int (сокращение от слова integer) и дробный float (числа с плавающей точкой). Над переменными можно выполнять различные операции, но прежде чем это делать, необходимо осуществить объявление и определение переменной.

Объявление переменной – это операция, при которой указывается тип переменной и её имя. Имя переменной может состоять из цифр и букв только латинского алфавита, а также символа подчёркивания, причём имя не должно начинаться с цифры. Регистр букв может быть любой (заглавные и строчные буквы) и имена переменных с одинаковым набором букв и с разным регистром будут восприниматься, как разные.

Определение переменной – это присвоение переменной некоторого значения. Определение может быть совмещено с объявлением, а может быть отдельно от него, но должно всегда следовать за объявлением. Иначе, компилятор выдаст ошибку.

```
//Объявление переменной x типа int
int x
//Определение переменной x значением 5
x
//Объявление переменной y типа float
float y
//Определение переменной y значением 1.589
y
//Объявление и определение переменной z типа int значением 6
int z
/*Объявление и определение переменной k типа float
значением переменной y */
float k = y;
```

Важно отметить, что для обозначения дробной части числа используется знак . («точка»), а не знак , («запятая»), как это обычно делается в математике.

5. Вывод числовой информации на экран контроллера

В большинстве случаев требуется вывести не только текстовую информацию, но и числа. Для этого также используется функция nxtDisplayTextLine, но с некоторыми особенностями. Она имеет вид:

```
nxtDisplayTextLine(1, "Value %d" , x);
```

Как видно, во втором аргументе функции появилось необычное обозначение %d, а также добавился третий аргумент функции. Обозначение %d означает, что мы хотим вывести число в виде целого знакового десятичного. Третий аргумент – это переменная, значение которой мы хотели

```
//Объявление и определение переменной
int x
//Вывод переменно в виде знакового целого числа

//Задержка отображения
wait1Msec(10000);
```

В результате работы программы на экран должно быть выведено

Value 5

Для того, чтобы вывести на экран число в виде десятичной дроби, необходимо использовать иной формат, который потребует замены

обозначения %d на %f. В итоге, программа будет выглядеть следующим

```
//Объявление и определение переменной
float x = 5.59
//Вывод переменно в виде знакового целого числа
nxtDisplayTextLine(1, "Value %f
//Задержка отображения
wait1Msec(10000);
```

В результате работы программы на экран должно быть выведено
Value 5.59

Задание 5.1

На основе приведённых примеров напишите программу, в которой осуществляется объявление трёх целочисленных переменных и присвоения им дня, месяца и года вашего рождения, а затем выведите информацию на экран, чтобы результат имел следующий вид

Day of birth 6
Month of birth 12
Year of birth 1980

6. Арифметические операции с переменными

Как и в математике, при программировании на языке С возможно выполнение арифметических операций с переменными. Рассмотрим некоторых из них.

1) *Сложение.* Для реализации операции сложения используется знак +,

```
//Объявление и определение переменных
int x = 5, y = 10, z
//Выполнение вычисления
z = x + y
//Выводим результат на экран контроллера
nxtDisplayTextLine(1, "Value %d" , z
//Задержка отображения
wait1Msec(10000);
```

В результате на экране будет отображено
Value 15

2) *Вычитание.* Для реализации операции вычитания используется знак –

```
//Объявление и определение переменных
int x = 5, y = 10, z
//Выполнение вычисления
z = x - y
//Выводим результат на экран контроллера
nxtDisplayTextLine(1, “Value %d” , z
//Задержка отображения
wait1Msec(10000);
```

В результате на экране будет отображено

Value -5

3) *Умножение.* Для реализации операции умножения используется знак

```
//Объявление и определение переменных
int x = 5, y = 10, z
//Выполнение вычисления
z = x * y
//Выводим результат на экран контроллера
nxtDisplayTextLine(1, “Value %d” , z
//Задержка отображения
wait1Msec(10000);
```

В результате на экране будет отображено

Value 50

4) *Деление.* Для реализации операции вычитания используется знак /. Поскольку результатом деления может быть дробное число, то используются переменные типа float,

```
//Объявление и определение переменных
float x = 5, y = 10, z
//Выполнение вычисления
z = x / y
//Выводим результат на экран контроллера
nxtDisplayTextLine(1, “Value %f” , z
//Задержка отображения
wait1Msec(10000);
```

В результате на экране будет отображено

Value 0.5

5) *Инкрементирование.* Инкрементирование – это операция увеличения на единицу для целого числа. Для реализации операции используется

```
//Объявление и определение переменных
int x = 5, y = x
//Выполнение вычисления
//Сначала выполняем традиционное сложение

//Затем выполняем операцию инкремента
y
//Выводим результат на экран контроллера
nxtDisplayTextLine(1, "Value x %d" , x);
nxtDisplayTextLine(2, "Value y %d" , y
//Задержка отображения
wait1Msec(10000);
```

В результате на экране будет отображено

Value x 6
Value y 6

6) *Декрементирование.* Декрементирование – это операция уменьшения на единицу для целого числа. Для реализации используется знак – –,

```
//Объявление и определение переменных
int x = 5, y = x
//Выполнение вычисления
//Сначала выполняем традиционное сложение
x = x -
//Затем выполняем операцию инкремента
y--
//Выводим результат на экран контроллера
nxtDisplayTextLine(1, "Value x %d" , x);
nxtDisplayTextLine(2, "Value y %d" , y
//Задержка отображения
wait1Msec(10000);
```

В результате на экране будет отображено

Value x 4
Value y 4

7) *Извлечение квадратного корня.* Для данной операции используется функция sqrt(), в качестве аргумента которой ставится число, из

```
//Объявление и определение переменных
float x = 9, z
//Выполнение вычисления
z = sqrt(x)
//Выводим результат на экран контроллера
nxtDisplayTextLine(1, "Value %f" , z
//Задержка отображения
wait1Msec(10000);
```

В результате на экране будет отображено

Value 3

8) *Возведение в степень.* Для этой операции используется функция pow(), в качестве аргументов используется два числа: первое – это то, которое нужно возвести в степень, а второе –

```
//Объявление и определение переменных
float x = 9, z
//Выполнение вычисления
z = pow(x,2)
//Выводим результат на экран контроллера
nxtDisplayTextLine(1, "Value %f" , z
//Задержка отображения
wait1Msec(10000);
```

В результате на экране будет отображено

Value 81

Задание 6.1

Напишите программу, которая выполняет вычисление выражения при заданных значениях x,y,z согласно номеру варианта, а затем выводит результат на экран:

1) r = (x + y*z) – 3.14159 * (x + y + z / x), при x = 10, y = 15, z = 3
2) r = (x – y*z) + 3.14159 * (x + y + z / y), при x = 11, y = 14, z = 5
3) r = (x + y*z) – 3.14159 * (x + y + z / x), при x = 12, y = 13, z = 7
4) r = (x – y*z) + 3.14159 * (x + y + z / y), при x = 13, y = 12, z = 11
5) r = (x + y*z) – 3.14159 * (x + y + z / y), при x = 14, y = 11, z = 13
6) r = (x – y*z) + 3.14159 * (x + y + z / x), при x = 15, y = 10, z = 17
7) r = (x + y*z) – 3.14159 * (x + y + z / y), при x = 16, y = 9, z = 19

8) r = (x – y*z) + 3.14159 * (x + y + z / x), при x = 17, y = 8, z = 23
9) r = (x + y*z) – 3.14159 * (x + y + z / y), при x = 18, y = 7, z = 29
10) r = (x – y*z) – 3.14159 * (x + y + z / x), при x = 19, y = 6, z = 31

7. Оператор условия

При реализации различных алгоритмов, как правило, возникают случаи, когда в зависимости от значения той или иной переменной требуется выполнить различный порядок действий. Самым простым примером может служить проверка, числа на положительность. Для реализации условия в языке С используется оператор if …. else

```
//Объявляем и определяем переменную
int x
/* Используя оператор условия, определяем,
число  на положительность */
if(x >
      //Если число положительно, то сообщаем об этом
      nxtDisplayTextLine(1, "Value is positive"
      //Задержка отображения
      wait1Msec(10000);

}
else
      //Если число отрицательно или равно 0, то сообщаем
      nxtDisplayTextLine(1, "Value is not
      //Задержка отображения
      wait1Msec(10000);

}
```

Оператор условия состоит из двух частей: в первой выполняется код в случае истинности условия (всё, что относится к if и стоит после него в скобках), а код второй части выполняется в том случае, когда условие не выполняется, то есть в противном случае. Наличие второй части не является обязательным.

8. Логические выражения

Алгебра логики – один из разделов математики, где рассматриваются операции с логическими выражениями. В области логических выражений переменные имеют два значения: истина или ложь, (1 или 0). Например, выражение $x > 0$ будет истинно при $x = 6$ и будет ложно при $x = -6$. Условия больше/меньше/равно являются одни из наиболее часто используемых, и для их реализации в языке С имеется форма записи, представленная в таблице

Оператор в языке С	*Аналог в математике*	*Название*
==	=	Равно
! =	≠	Не равно
>	>	Больше
<	<	Меньше
>=	≥	Больше или равно
<=	≤	Меньше или равно

Используя такие обозначения в операторе условия if можно составлять различные комбинации. Например, напишем программу, выполняющую

```
//Объявляем и определяем переменные
float x = 8.56, y = 6.35, z
//Мы собираемся разделить x на y
//Поэтому проверяем, не равен ли y нулю
if(y
      //Выполняем деление
      z = x/y
      //Выводим значение
      nxtDisplayTextLine(1, "Result %f" , z
      //Задержка отображения
      wait1Msec(10000);


//В противном случае выводим ошибку
else {
      nxtDisplayTextLine(1, "Error: y
      //Задержка отображения
      wait1Msec(10000);

}
```

Задание 8.1

1. Пусть дан круг радиуса R и квадрат со стороной 6. Нужно написать программу, которая определяет, помещается ли круг с заданным радиусом в данный квадрат (диаметр круга должен быть меньше стороны квадрата). Значение R задаёт преподаватель.
2. Пусть дан квадрат со стороной L и круг радиусом 6. Нужно написать программу, которая определяет, помещается ли квадрат с заданной стороной в круг (диаметр круга должен быть больше диагонали квадрата, которая находится по теореме Пифагора). Значение L задаёт преподаватель

3. Пусть дана окружность с радиусом R и равносторонний треугольник с длиной стороны 6. Написать программу, которая определяет, поместится ли треугольник внутри окружности (радиус описанной вокруг равностороннего треугольника окружности находится как $R = \frac{a}{\sqrt{3}}$, где a – длина стороны треугольника). Значение R задаёт преподаватель.
4. Пусть дана окружность радиусом 11 и равносторонний треугольник с длиной стороны a. Написать программу, которая определяет, поместится ли данная окружность в треугольник (радиус вписанной окружности в равносторонний треугольник находится по формуле $R = \frac{a}{2\sqrt{3}}$). Значение a задаёт преподаватель.
5. Пусть дана окружность с радиусом R и правильный пятиугольник с длиной стороны 5. Написать программу, которая определяет, поместится ли пятиугольник внутри окружности (радиус описанной вокруг правильного пятиугольника окружности находится как $R = \frac{\sqrt{10}\sqrt{5+\sqrt{5}}}{10} \cdot a$, где a – длина стороны пятиугольника). Значение R задаёт преподаватель.
6. Пусть дана окружность радиусом 9 и правильный пятиугольник с длиной стороны a. Написать программу, которая определяет, поместится ли данная окружность в пятиугольник (радиус вписанной окружности в правильный пятиугольник находится по формуле $R = \frac{\sqrt{5}\sqrt{5+2\sqrt{5}}}{10} \cdot a$). Значение a задаёт преподаватель.
7. Пусть дана окружность с радиусом R и правильный шестиугольник с длиной стороны 8. Написать программу, которая определяет, поместится ли шестиугольник внутри окружности (радиус описанной вокруг правильного шестиугольника окружности равен его стороне). Значение R задаёт преподаватель.
8. Пусть дана окружность радиусом 8 и правильный шестиугольник с длиной стороны a. Написать программу, которая определяет, поместится ли данная окружность в шестиугольник (радиус вписанной окружности в правильный шестиугольник находится по формуле $R = \frac{\sqrt{3}}{2} \cdot a$). Значение a задаёт преподаватель.
9. Пусть дана окружность с радиусом R и правильный восьмиугольник с длиной стороны 7. Написать программу, которая определяет, поместится ли восьмиугольник внутри окружности (радиус описанной вокруг правильного восьмиугольника окружности находится как $R = a \cdot \sqrt{\frac{1+\sqrt{2}}{\sqrt{2}}}$). Значение R задаёт преподаватель.
10. Пусть дана окружность радиусом 25 и правильный восьмиугольник с длиной стороны a. Написать программу, которая определяет, поместится ли данная окружность в восьмиугольник (радиус

вписанной окружности в правильный восьмиугольник находится по формуле $R = \frac{1+\sqrt{2}}{2} \cdot a$). Значение а задаёт преподаватель.

9. Составление комбинированных логических выражений

При использовании оператора ветвления иногда возникает необходимость проверки комбинированного условия. Для этого могут быть применены классические операции алгебры логики. Рассмотрим их представление в языке программирования C/C++.

1. Отрицание или инверсия. Операция меняет значение выражения на обратное. В алгебре логики принято обозначать эту операцию как черту над аргументом, например $\overline{A}$. Рассмотрим действие этого выражения на примере высказывания:

A : Мальчик купил хлеб.

Обратное к этому высказыванию будет звучать как

$\overline{A}$ *: Мальчик не купил хлеб.*

Таким образом, всё что было истинным станет ложным, а всё, что было ложным станет истинным. Таблица истинности выглядит следующим образом

A	$\overline{A}$
0	1
1	0

В языке C для обозначения операции используется восклицательный знак, например

```
!(x > 0)
```

2. Дизъюнкция или логическое сложение, также операция ИЛИ. Операция применяется к двум логическим переменным или выражениям, результат является истинным если хотя бы один из аргументов имеет истинное значение. В алгебре логики обычно обозначается как $A \cup B$. Пусть имеются два высказывания:

A : *Мальчик купил чёрный хлеб.*

B : *Мальчик купил белый хлеб.*

Операция ИЛИ, применённая к этим выражениям будет выглядеть как

$A \cup B$: *Мальчик купил чёрный или белый хлеб.*

Таблица истинности выглядит следующим образом

A	B	$A \cup B$
0	0	0
1	0	1

0	1	1
1	1	1

В языке С для обозначения этой операции применяется знак двойной вертикальной черты, например

```
x > 0 || y > 0
```

3. Конъюнкция или логическое умножение, также операция И. Операция применяется к двум логическим аргументам или выражениям, результат является истинным, если все аргументы имеют истинное значение. Если хотя бы один из них имеет значение «ложь», то и результат будет ложным. В алгебре логики операция обычно обозначается как $A \cap B$. Пусть имеются два высказывания:

A: *Мальчик купил чёрный хлеб.*

B: *Мальчик купил белый хлеб.*

Операция И, применённая к этим выражениям будет выглядеть как

$A \cup B$: *Мальчик купил чёрный и белый хлеб.*

Таблица истинности выглядит следующим образом

A	B	$A \cap B$
0	0	0
1	0	0
0	1	0
1	1	1

В языке С для обозначения этой операции применяется знак двойного амперсанда, например

```
x > 0 && y > 0
```

Таким образом, применяя эти три операции, возможно комбинирование логических условий для получения необходимой логической последовательности.

10. Циклы. Цикл с параметром

Цикл – это многократное повторение одинаковых операций для определённых объектов. Цикл может быть использован в тех случаях, когда для однотипных объектов нужно последовательно выполнить ряд действий, например, в случае с роботом это может быть проверка всех моторов и вывод сообщения об их состоянии. Выделяют циклы с параметром и циклы без параметра

Циклы с параметром используются в тех случаях, когда число повторений операции заранее известно. Каждое повторение цикла называется итерацией.

Для реализации цикла с параметром в языке С используется оператор for.

```
//Объявляем и определяем переменную
int x
//Используем цикл с параметром
for(int i = 1; i < 10; i
      //На каждой итерации умножаем x на 2
      x = x
      //Выводим получаемый результат
      nxtDisplayTextLine(1, "Result %d %f" , i, x
      //Задержка отображения
      wait1Msec(3000);

}
```

Изначально осуществляется объявление и определение переменной x. Начальное определение переменной называется инициализация данной переменной. Переменная была проинициализирована значением 1. Далее идёт цикл с параметром. У цикла for есть три параметра, они перечислены через точку с запятой внутри скобок:

1. Объявление переменной цикла (часто называемой счётчиком), которая отвечает за номер текущей итерации. При объявлении также осуществляется инициализация переменной начальным значением диапазона, который будет проходить цикл. Рекомендуется, чтобы переменная была целочисленной и в данном случае она инициализируется единицей.
2. Условие выполнения цикла. Цикл будет выполняться до тех пор, пока не будет достигнуто такое значение переменной i, при котором данное условие становится ложным.
3. Шаг цикла. На каждой итерации цикла значение переменной i будет изменяться согласно приведённому выражению.

Итак, анализируя имеющиеся здесь параметры можно сказать, что для цикла в приведённом примере начальным значением является единица, на каждой итерации значение переменной, отвечающей за номер будет увеличиваться на единицу и так будет продолжаться до тех пор, пока значение достигнет 9. При следующем значении (10) условие в параметре 2 уже не будет выполняться, поэтому выполнение цикла прекратится. На каждой итерации цикла выполняется умножение на 2 имеющего значения x, таким образом, мы получаем программу, которая последовательно вычисляет числа, являющиеся степенью числа 2 и выводит их на экран.

Задание 10.1

1. Написать программу с циклом с параметром, которая последовательно выводит число 3 в степени от 1 до 9.

2. Написать программу с циклом с параметром, которая последовательно выводит число 4 в степени от 8 до 1.
3. Написать программу с циклом с параметром, которая последовательно выводит нечётные числа от 1 до 23.
4. Написать программу с циклом с параметром, которая последовательно выводит нечётные числа от 25 до 3.
5. Написать программу с циклом с параметром, которая последовательно выводит чётные числа от 2 до 24.
6. Написать программу с циклом с параметром, которая последовательно выводит чётные числа от 22 до 4.
7. Написать программу с циклом с параметром, которая последовательно выводит число 5 в степени от 1 до 5
8. Написать программу с циклом с параметром, которая последовательно выводит число 5 в степени от 5 до 1.
9. Написать программу с циклом с параметром, которая последовательно выводит число 6 в степени от 2 до 8.
10. Написать программу с циклом с параметром, которая последовательно выводит число 6 в степени от 7 до 3.

11. Циклы с предусловием

В тех случаях, когда количество итераций цикла заранее неизвестно, следует использовать цикл без параметра. Цикл без параметра делится на цикл с предусловием и цикл с постусловием.

В цикле с предусловием на каждой итерации проверяется значение переменной цикла. Если это значение не удовлетворяет условию, то осуществляется выход из цикла. При этом, цикл может не выполниться ни разу, если на входе в цикл переменная уже не удовлетворяет заданному условию

```
//Инициализация переменной цикла
int i
//Инициализация рабочей переменной
int x
//Запускаем цикл с предусловием
while(i
      //Инкрементируем переменную цикла
      i
      //На каждой итерации умножаем x на 2
      x = x
      //Выводим получаемый результат
      nxtDisplayTextLine(1, "Result %d %f" , i, x
      //Задержка отображения
      wait1Msec(3000);
}
```

Для реализации цикла с предусловием используется оператор while. Параметром является некое условие (в данном случае то, что переменная цикла меньше 10), а сам параметр объявляется и инициализируется перед циклом. В теле цикла происходит изменение параметра заданным образом, в данном случае его инкрементирование.

Задание 11.1

Согласно варианту выполнить задание, аналогичное представленному в пункте 10, но с использованием цикла с предусловием.

12. Цикл с постусловием

Цикл с постусловием также используется в тех случаях, когда количество итераций заранее неизвестно, однако, в отличии от цикла с предусловием условие выполнения цикла проверяется по окончанию итерации, что приводит к тому, что тело цикла выполнится хо

```
//Инициализация переменной цикла
int i
//Инициализация рабочей переменной
int x
//Запускаем цикл с постусловием
do

      //Инкрементируем переменную цикла
      i
      //На каждой итерации умножаем x на 2
      x = x
      //Выводим получаемый результат
      nxtDisplayTextLine(1, "Result %d %f" , i, x
      //Задержка отображения
      wait1Msec(3000);
}
while(i < 10);
```

Как видно из примера, в текст программы добавляется служебное слово do, после которого следует тело цикла. По окончанию цикла осуществляется проверка условия, в результате которой цикл либо завершает свою работу, либо продолжает выполнять итерации.

Задание 12.1

Согласно варианту выполнить задание, аналогичное представленному в пункте 10, но с использованием цикла с постусловием.

13. Оператор переключения switch

В некоторых ситуациях выбора одного условия может быть недостаточно. Представим себе светофор, причём рабочий. На рисунке 1 показаны несколько вариантов светофора. Для пешехода, который переходит улицу, можно использовать простой светофор, такой, который изображён на рисунке 1-в.

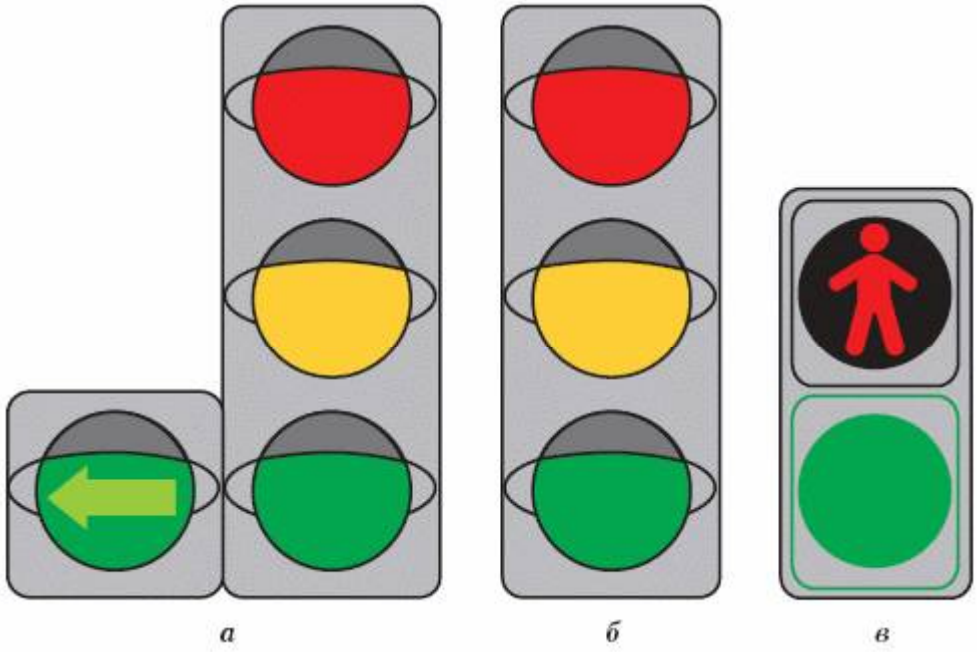

Рисунок 1 – Различные варианты светофора

На светофоре для пешехода есть всего два положения: горит зелёный или горит красный (случай моргания зелёного мы не рассматриваем). Поэтому, подойдя к регулируемому пешеходному переходу, человек может либо перейти улицу, если горит зелёный, либо стоять и ждать, если горит красный. Если мы заходим написать программу, то нам будет достаточно одной структуры if/else. Условно программа показана ниже

```
if(TrafficLights == "GREEN
      //Переходим улицу
}
else
      //Стоим и ждём
}
```

Однако, светофор для автомобиля несколько отличается, и самым главным отличием является наличие жёлтого сигнала (рисунок 1-б). Кроме этого, на некоторых светофорах есть дополнительная стрелка, которая разрешает поворот (рисунок 1-а). Попробуем написать программу с использованием оператора условия для перекрёстка с таким светофором:

```
if(TrafficLights == "RED" {
```

```
        //Свет красный, стоим и ждём
    }

        //Свет красный с жёлтым, подготавливаемся к началу
движения, но всё ещё ждём
    }
    if(TrafficLights == "GREEN MAIN
        //Зелёный свет для движения прямо, можем пересекать
перекрёсток только в прямом направлении
    }
    if(TrafficLights == "YELLOW
        //Останавливаемся, выезжать на перекрёсток уже нельзя,
но если выехали - нужно его пересечь
    }
    if(TrafficLights == "RED AND ARROW LEFT
        //Горит стрелка налево и основной красный свет - прямо
движение запрещено, но налево можно
    }
    if(TrafficLights == "GREEN AND ARROW LEFT
        //Можно ехать и прямо и налево
    }
```

Итак, как видно, множество условий if позволяют описать поведение. Однако, программа при этом выглядит громоздко, и может быть непонятна при разборе. Поэтому, в языке С имеется оператор выбора switch/case. Он исследует значение переменной и при соответствующих значениях выполняет соответствующие действия. Перепишем код для светофора, представленный

```
    //Оператор switch проверяет значение переменной
    switch(TrafficLights)
        //Затем для каждого случая выполняются действия
        case "RED
            //Свет красный, стоим и ждём
        break;
        case "RED AND YELLOW
            //Свет красный с жёлтым, подготавливаемся к началу
движения, но всё ещё ждём
        break;
        case "GREEN MAIN"
            //Зелёный свет для движения прямо, можем пересекать
перекрёсток только в прямом направлении
        break;
        case "YELLOW"
            //Останавливаемся, выезжать на перекрёсток уже
нельзя, но если выехали - нужно его пересечь     break;
        case "RED AND ARROW LEFT"
            //Горит стрелка налево и основной красный свет -
прямо движение запрещено, но налево можно
        break;
        case "GREEN AND ARROW LEFT":
```

```
            //Можно ехать и прямо и налево
        break
        //Значение по умолчанию
        default
            //Оставаться на месте
        break;
}
```

В начале программы осуществляется проверка переменной. Далее, в теле оператора switch имеется множество операторов case, каждый из который выполняется только в том случае, если значение переменной точно удовлетворяет значению, стоящему после оператора. Далее ставится двоеточие и перечисляются действия, которые необходимо выполнить в этом случае.

По окончанию выполнения действий в каждом случае осуществляется остановка служебным словом break. После этого завершается функционирование данного оператора и программа следует дальше.

Особое внимание стоит обратить на служебное слово default. Оно описывает случай по умолчанию, то есть при котором ни одно из указанных условий не выполняется. Например, светофор закрыл проезжающий грузовик и мы не можем получить его значение. В этом случае мы не можем предпринять каких то действий и стоим на месте, ожидая появления информации.

Приведём другой пример. Пусть имеется программа, которая считает значение некоторой функции, причём используя оператор switch. Значения

```
//Значение функции
int y
//В цикле проходим по значениям от 1 до 5
for(int x = 1; x < 6; x
      //С использованием оператора switch вычисляем
      switch(x)
      //Затем для каждого случая выполняются действия
            case 1:
                  y = 2;
            break;
            case 2:
                  y = 4;
            break;
            case 3:
                  y = 6;
            break;
            case 4:
                  y = 4;
            break;
            case 5:
                  y = 2;
            break;
```

```
        //Выводим изображение на экран

        //Вводим задержку
        wait1Msec(1000);
}
```

Задание 13.1

Напишите программу с использованием switch/case, которая последовательно в качестве значения выводит цифру года вашего рождения, например для 1980 будет

1
9
8
0

14. Массивы

Массив – это множество однотипных значений, хранящихся в одной переменной. Массивы используются в тех случаях, когда имеющееся множество элементов достаточно большое, чтобы обозначать каждый элемент отдельной переменной. Например, в качестве массива можно представить список учеников класса, или список деталей, входящих в набор.

Массив обозначается квадратными скобками с указанием количества элементов. Например:

```
int list[5];
```

Нами был создан массив, состоящих из 5 элементов типа int. При объявлении массива мы должны точно знать количество элементов, входящих в его состав. При работе программы это количество нельзя превышать, иначе будет ошибка.

Обращение к элементу массива осуществляется с указанием его номера. Стоит помнить, что нумерация элементов массива в языке С начинается с 0, то

```
//Обращение к первому элементу и присвоение значения
list
//Обращение к третьему элементу и присвоение значения
list[2] = 5;
```

Обращение к элементам может быть осуществлено в цикле. Ниже показан пример программы, которая сначала считает квадрат чисел, записывая их в

```
//Объявляем массив
int square[10];
```

```
//В цикле с параметром присваиваем значение квадрата
for(int i=0; i < 10; i
      //Вычисляем квадрат параметра цикла и записываем его в
      //массив
      square[i] = i*i;

 //Выводим в цикле последовательно на экран массив

     //Выводим строчку

     //Задержка
     wait1Msec(1000);
 }
```

Сначала в программе объявляется массив, состоящих из 10 элементов. Затем, каждому элементу этого массива присваивается квадрат номера этого элемента. Всё это осуществляется в цикле с параметром. Затем, в другом цикле с параметром осуществляется последовательный вывод элементов.

Задание 14.1

Напишите программу, которая создаёт массив из трёх элементов, записывает туда данные о дне, месяце и годе вашего рождения, а затем выводит эту информацию на экран.

Задание 14.2

Напишите программу, которая использует массив, состоящий из количества элементов, равного номеру вашего варианта, умноженного на 3 и вычисляет для каждого элемента квадратный корень из номера этого элемента, а затем выводит получившееся значение на экран.

15. Функции

Функция – совокупность операций, объединенных в программном коде, которые, как правило, выполняют некоторую задачу с определёнными входными и выходными данными. Функцию можно представить как блок следующего вида

Функция может многократно вызываться в течении программы, таким образом избавляя разработчика от необходимости постоянно вставлять необходимый для выполнения код. В качестве входных данных используются те значения, которые необходимы для выполнения операций функции, они

носят название *аргументы функции*. Выходные данные – это результат работы функции, которые в программе на языке С чаще всего представлены в виде *возвращаемого значения*. Идентификатором функции является её *название*. Рассмотрим пример задачи, которую можно реализовать с использованием функции.

Пусть даны длины трёх сторон треугольника и необходимо определить, может ли существовать такой треугольник. Существование треугольника можно проверить исходя из условия, что сумма двух сторон больше третьей. То есть необходимо проверить истинность трёх неравенств. Напишем

```
    /* Объявляем три переменные, в которых будут храниться длины
сторон */

        //Задаём значения
        a_tr_1 = 3;
        b_tr_1 = 4;

        //Проверяем условие
        if((a_tr_1 + b_tr_1 > c_tr_1) && (a_tr_1 + c_tr_1 > b_tr_1)
&& (b_tr_1 + c_tr_1 > a_tr_1)) {
            nxtDisplayTextLine(1, "Triangle exists");
        }
        else {
            nxtDisplayTextLine(1, "Triangle does not exist");
        }
        return 0;
```

В программе используется достаточно большое составное условие. Для одного треугольника программа выглядит небольшой, однако если необходимо проверить таким образом несколько треугольников, то следует повторить написание условия многократно. Такой подход непродуктивен – вместо этого можно реализовать функцию, которая будет осуществлять проверку существования треугольника. К функции будут предъявляться следующие требования: три входных параметра – аргументы функции типа float – длин сторон треугольника, один выходной параметр – возвращаемое значение типа bool – существует (значение «истина») или нет (значение «ложь») треугольник с указанными параметрами, название функции triangleExistance, которое характеризует задачу функции. Исходный код примера программы пр

```
    //Объявление и определение функции
    bool triangleExistance(float a, float b, float c
        //Переменная, отвечающая за результат

        //Проверка
        if((a + b > c) && (a + c > b) && (b + c> a)) {
            result = true;
```

```
        }
        else {
            result = false;

        //Возврат результата
        return result;
    }
    task main()

        /* Объявляем три переменные, в которых будут хранится длины
сторон */
        float a_tr_1, b_tr_1, c_tr_1;
        //Задаём значения
        a_tr_1 = 3;
        b_tr_1 = 4;

        //Проверяем условие
        if(triangleExistance(a_tr_1, b_tr_1, c_tr_1)) {
            nxtDisplayTextLine(1, "Triangle exists");
        }
        else {
            nxtDisplayTextLine(1, "Triangle does not exist");
        }
        return 0;
    }
```

Рассмотрим синтаксис созданной функции.

Тип возвращаемого значения	Название функции	Аргументы функции
bool	triangleExistance	(float a, float b, float c)

Первым элементом функции является тип возвращаемого значения. Функция может возвращать таким образом либо одно значение, либо вовсе не возвращать ни одного значения. Далее указывается название функции – её уникальный идентификатор, который позволит вызвать эту функцию. Далее в круглых скобках указываются аргументы функции. Если их несколько, то их перечисление осуществляется через запятую. Даже если аргументы имеют одинаковый тип, то он всё равно указывается для каждого аргумента. Тело функции располагается между двумя фигурными скобками.

В теле функции объявляется переменная result, которая имеет тип bool. Значение этой переменной определяется в зависимости от выполнения или невыполнения условия, которое проверяется далее. Значение переменной result возвращается с помощью оператора return.

В программе вызов функции осуществляется непосредственно в операторе ветвления, таким образом, в зависимости от значения результата будет выведено одно из двух сообщений.

Прежде чем использовать функцию в программе, её необходимо объявить. Однако ввиду того, что количество операций в теле функции может быть достаточно большим, принято разделять объявление и определение функции, причём определение функции осуществлять после функции main. Изменим код программы из примера с треугольником, разделив объявление и

```
//Объявление функции
bool triangleExistance(float a, float b, float c);
task main()

    /*Объявляем три переменные, в которых будут хранится длины
сторон*/

    //Задаём значения
    a_tr_1 = 3;
    b_tr_1 = 4;

    //Проверяем результат, который возвратит функция
    if(triangleExistance(a_tr_1, b_tr_1, c_tr_1)) {
        nxtDisplayTextLine(1, "Triangle exists");
    }
    else {
        nxtDisplayTextLine(1, "Triangle does not exist");
    }
    return 0;

//Определение функции
bool triangleExistance
    //Переменная, отвечающая за результат
    bool result = (a + b > c) && (a + c > b) && (b + c> a
    //Возврат результата
    return result;
}
```

Объявление функции также часто называют прототипом функции. Поскольку функция main() вызывается первой, она не должна быть похоронена в исходном коде функций, объявленных до неё, вследствие чего использование прототипов является предпочтительным вариантом.

В зависимости от задач, выполняемых функцией, выделяют несколько основных вариантов:

1. *Функция имеет одно возвращаемое значение и один или несколько аргументов.* Примером такой функции является

```
bool triangleExistance(float a, float b, float c);
```

которая была рассмотрена ранее. Функции такого вида являются классическими и достаточно часто используются в программах.

2. *Функция не имеет возвращаемого значения, но имеет один или несколько аргументов.* Для обозначения отсутствия возвращаемого значения используется ключевое слово void при указании типа возвращаемого значения. Такие функции могут использоваться для выполнения операций над переменными, не относящимся к тому блоку кода, откуда функции вызывается. Рассмотрим, пример такой функции, которая выводит экран

```
//Объявление и определение функции
void symbolLine(char symbol, int n) {
    for(int i = 0; i < n; i++) {
        nxtDisplayTextLine(1, "%c", symbol);
    }
}
task main()

    //Вызов функции
    symbolLine('*', 20);
    return 0;
}
```

В этой программе используется переменная типа char. она предназначена для хранения одного символа и в данном случае в качестве этого символа указываем '*'. Не следует путать с операцией умножения.

3. *Функция имеет возвращаемое значение, но не имеет аргументов.* Такие функции, как правило, используются для получения вызывающей частью программы некоторого значения, которое может быть получено извне. Рассмотрим пример такой программы, которая имеет функцию, возвращающее значение числа П

```
//Объявление и определение функции
float getPi() {
    return 3.14159;
}
task main()
{
    float p = getPi();
    nxtDisplayTextLine(1, "Pi = %f", p);
    return 0;
}
```

Стоит отметить, что несмотря на отсутствие аргументов, круглые скобки после названия функции всё равно указываются – это один из отличительных признаков функции.

4. *Функция не имеет возвращаемого значения и не имеет аргументов.* Такие функции применяются, когда необходимо выполнить действие,

независящее от каких либо факторов. Например, рассмотрим функцию, которая выводит линию из 1

```
//Объявление и определение функции
void starLine() {
    for(int i = 0; i < 10; i++) {
        nxtDisplayTextLine(1, "%c", symbol);
    }
    cout<<endl;
}
task main()

    //Вызов функции
    starLine();
    return 0;
}
```

Таким образом, использование функций позволяет упростить программу и сократить объём используемого кода.

Задание 15.1

Напишите функцию, которая выводит значения согласно заданию в разделе 10.

Содержание