Урок 3. Условие

В первых двух уроках Вы ознакомились с выводом на экран, вводом с клавиатуры и вычислением простейших арифметических выражений. Но все наши программы объединяло то, что в них менялись только числа на входе, а ход выполнения программы от входных данных не зависит.

На этом уроке Вы познакомитесь с условным оператором, который позволяет изменять ход выполнения программы и определять, какие именно строки будут выполняться. Изучать будем как всегда на примерах.

Для каждого задания заведите отдельный проект в отдельной папке. Например, для задания 1 из урока 3 – проект `lesson3_1`, файл с текстом программы – `lesson3_2.c`. Для второго задания – `lesson3_2` и `lesson3_2.c` и т.д.

Задание 1.

Программа спрашивает пароль (целое число), и в случае совпадения выводит строку «Добро пожаловать!»

Решение

```
#include <stdio.h>

void main()
{
    int pwd, a;

    pwd = 1024;
    printf("Введите пароль (целое число): ");
    scanf("%d", &a);

    if (a == pwd)
    {
        printf("Добро пожаловать!");
    }

    printf("\n"); while (!kbhit());
}
```

Всё, кроме операторов `if` и `==` уже встречалось ранее. Но на всякий случай напомним смысл строк:

```
int pwd, a;
pwd = 1024;
```

Сначала объявляются переменные pwd и a, а затем в переменную pwd помещается правильный пароль (1024, можете заменить на другое целое число).

```
printf("Введите пароль (целое число): ");
scanf("%d", &a);
```

Здесь выводится приглашение к вводу пароля, а затем пароль вводится с клавиатуры и попадает в переменную `a` (знак & перед её именем означает, что функции scanf передаётся адрес ячейки `a` в памяти компьютера)

```
if (a == pwd)
{
  printf("Добро пожаловать!");
}
```

Но по настоящему новым здесь является **условный оператор** `if` (переводится как «Если»). Он работает так:

1. Проверить истинность условия в круглых скобках
2. **Если условие истинно –** выполнить часть программы в фигурных скобках после оператора `if`
3. **Если условие ложно –** пропустить часть программы в фигурных скобках после оператора `if`

Сдвоенный знак равенства `==` – это **оператор сравнения**, который проверяет, равны ли значения переменных (числа в ячейках) a и pwd. Если они равны, то условие истинно, пароль правильный и на появится надпись «Добро пожаловать!» **Пробелы между двумя знаками равенства в операторе сравнения делать нельзя.**

Внимание: ни в коем случае не путайте операторы == и =. Первый – оператор сравнения, второй – оператор присваивания.

Упражнение: замените в этой программе знак == на =, откомпилируйте и запустите её. Попробуйте ввести как верный, так и неверный пароль. И программа будет говорить «Добро пожаловать» на любой пароль (ведь присваивание `a=pwd` будет всегда выполняться и возвращать значение, присвоенное `a`. Т.к. это значение ненулевое (`1024`), то условие будет всегда истинным).

Задание 2.

Модифицируйте предыдущую программу так, чтобы в случае ввода неверного пароля она писала «Неверный пароль»

Решение:

```
#include <stdio.h>

void main()
{
  int pwd, a;

  pwd = 1024;
  printf("Введите пароль (целое число): ");
  scanf("%d", &a);

  if (a == pwd)
  {
    printf("Добро пожаловать!");
  }
  else
  {
    printf("Неверный пароль!");
  }

  printf("\n"); while (!kbhit());
}
```

Здесь тоже использован условный оператор, но со словом `else` (переводится как «Иначе»). Он работает почти так же:

1. Проверить истинность условия в круглых скобках
2. **Если условие истинно**, **то** выполняется содержимое фигурных скобок сразу после `if`
3. **Если условие ложно – то** выполняется содержимое скобок после `else`

Те, кто ввёл верный пароль, будут по-прежнему читать приветствие, а не знающие пароля – сообщение о неверном пароле.

Задание 3.

Напишите программу, которая сравнивает два числа и может вывести три результата: a<b, a>b и a=b. Должны поддерживаться десятичные дроби.

Решение:

```
#include <stdio.h>

main()
{
  double a, b;

  printf("Программа сравнивает два числа\n");
  printf("Введите число a: ");
  scanf("%lf", &a);
  printf("Введите число b: ");
  scanf("%lf", &b);

  if (a == b)
  {
    printf("a = b");
  }
  else if (a > b)
  {
    printf("a > b");
  }
  else
  {
    printf("a < b");
  }

  printf("\n"); while (!kbhit());
}
```

Здесь появляется сочетание слов `else if`. Если первое условие (после `if`) не выполняется, тогда будет проверено второе условие (после `else if`). Если оно выполняется, то управление получит вторая группа операторов. А если нет – то начнут выполняться содержимое фигурных скобок после `else`.

Операторы сравнения в языке Си

Язык Си	Математика
==	= (равно)
<	< (меньше)
>	> (больше)
<=	≤ (меньше или равно)
>=	≥ (больше или равно)
!=	≠ (не равно)

Задание 4.*

Напишите программу для решения квадратного уравнения. Рассмотрите три случая: два корня (D > 0), один корень (D = 0) и корней нет (D < 0).

Формулы для решения квадратного уравнения

$$D = b^2 - 4ac;\ x_{1,2} = \frac{-b \pm \sqrt{D}}{2a}$$

Решение:

```
#include <stdio.h> // Стандартная библиотека
#include <math.h>  // Математическая библиотека

void main()
{ /* Объявление переменных */
  double D, a, b, c, x1, x2;
  /* Вывод подсказки */
  printf("Программа решает квадратное уравнение\n");
  printf("ax^2 + bx + c = 0\n");
  /* Ввод коэффициентов */
  printf("Введите a: ");
  scanf("%lf", &a);
  printf("Введите b: ");
  scanf("%lf", &b);
  printf("Введите c: ");
  scanf("%lf", &c);
  /* Расчёт дискриминанта */
  D = b*b - 4*a*c;
  printf("Дискриминант: %lf\n", D);
  /* Расчёт корней
     и их вывод на экран */
  if (D > 0)        // Положительный дискриминант
  {
    x1 = -b+sqrt(D) / (2*a); // sqrt(D) - это
    x2 = -b-sqrt(D) / (2*a); // квадратный корень из D
                             // (функция из math.h)
    printf("Два корня: %lf и %lf", x1, x2);
  }
  else if (D == 0) // Нулевой дискриминант
  { x1 = -b / (2*a);
    printf("Один корень: %lf", x1);
  }
  else             // Отрицательный дискриминант
  {
    printf("Корней нет!");
  }

  printf("\n"); while (!kbhit());
}
```

Пояснения к тексту программы

```
#include <math.h>  // Математическая библиотека
```

Эта строка подключает математическую библиотеку к программе. В ней есть функции для вычисления квадратного корня, тригонометрические функции и т.п.

Сдвоенная косая черта `//` - начало **комментария**. Всё, что написано между ним и концом строки, пропускается при выполнении программы. `//` обычно используют для коротких однострочных комментариев.

```
/* Расчёт корней
   и их вывод на экран */
```

Это второй способ написать комментарий. Всё, что находится между `/*` и `*/` также пропускается компилятором. Это очень удобно для многострочных и подробных комментариев.

```
D = b*b - 4*a*c;
```

Вычисление дискриминанта. В языке Си приоритет действий здесь соблюдается автоматически.

```
x1 = -b+sqrt(D) / (2*a);
```

Вычисление одного из корней уравнения. Функция `sqrt` вычисляет квадратный корень вещественного (дробного) числа.

Упражнения

1. Напишите программу, которая различает отрицательное и положительное число. Предусмотрите также реакцию на 0.
2. Напишите программу, которая сначала запрашивает угол в градусах, а затем определяет, какой он - острый, тупой или прямой.
3. Определить, чётное или нечётное число.
   Указание: используйте выражение `a % 2` для того, чтобы получить остаток от деления a на 2. Для переменных используйте целочисленный тип `int`.
4. Даны два целых числа – a и b. Нужно определить, делится ли a на b.
   Указание: используйте выражение `a % b` для того, чтобы получить остаток от деления a на b.
5. Напишите программу, которая сравнивает две денежные суммы в разных валютах (например, долларах и евро). Она должна запрашивать сами суммы и курсы двух валют в рублях.
   Указание: переведите каждую из сумм в рубли, а уже затем сравнивайте. Если одна из сумм – в рублях, то в качестве курса введите единицу.