Управляющие конструкции: ветвления и переходы. Тернарный оператор. Конструкция switch-case. Примеры

Операторы ветвления
Операторы ветвления позволяют создавать условие, при разных исходах которого будет выполнятся разный код.

Оператор `if-else`
Основным оператором ветвления является `if-else`, он включает в себя два блока кода, которые выполняются при разных исходах условия.

- Синтаксис

```
if (a == 5)
{
          //один код
}
else
{
          // другой код
}
```

  Как и в других языках ветка `else` не является обязательной.
  Также, после `if` можно использовать `else if`, чтобы явно задать другое условие для выполнения какого либо кода:

```
if(a == 5)
{
        //делать одно
}
else if(a == 10)
{
        // делать второе
```

```
}
else
{
        //делать другое
}
```

Количество ветвлений `else if` никак не ограничено, можно сделать хоть миллион таких условий (но всё же сильно много делать не надо, лучше рассмотреть другой вариант).

Тернарный оператор
Тернарное выражение, грубо говоря, представляет из себя `if-else` в одну строку, то бишь, позволяет сократить простые условия до одной строки и записать результат в переменную.
Тернарное выражение возвращает одно из двух заданных выражений.

- Синтаксис

```
int x = 12;
int y = 16;
int z = (x > y) ? x : y;
```

1. `(x > y)` - некоторое условие.
2. `x` - первое выражение.
3. `y` - второе выражение.
4. `?` - стоит между условием и выражениями, дабы отделить их друг от друга.
5. `:` - Разделитель между двумя выражениями. Возвращается либо первое, либо второе, никак иначе.
   Что тут происходит? Лично я изначально научился читать эти выражения так, используя, так сказать, знак вопроса:
   `x` больше чем `y`? Если ДА, то возвращаем (то бишь записываем в переменную) значение `x`, если НЕТ - записываем `y`.
   В прочем, к этому просто стоит привыкнуть.
   Стоит заметить, что в тернарном выражении нельзя создавать сложные "тела", и все должно влезть в одну строку.

Конструкция `switch-case`
`switch` используется, когда появляется необходимость множественного выбора в

зависимости от какой либо переменной, дабы избежать кучи `if-else`.

- Синтаксис

```
switch(a)
{
        case 0:
                // Код, в случае если a = 0
                break;
        case 1:
                //Код в случае если a = 1
                break;
        //...
        default:
                //Код, если ни одно условие не подошло.
}
```

Вместо `a` пишем ту переменную, значение которой будем сравнивать, а внутри фигурных скобок описываем разные случаи, чему равна эта переменная. Оператор `break` стоит писать всегда, в противном случае, если совпадение найдется, помимо необходимого `case` выполнятся и те, что стоят после него. Ветка `default` не обязательна.

Плохой `goto`
Помимо привычных операторов ветвления есть еще тот, чье имя нельзя ~~произносить~~ использовать - это `goto`. Он плох тем, что создает полнейшую неразбериху в коде, как визуально, так и на уровне компиляции.
Данный пример стоит просто запомнить, просто для того, чтобы знать что такое есть. Не стоит это использовать.

```
goto jump_here;
// ...
jump_here:
        printf("goto - bad");
        return 0;
```

Создается некоторое место, где выполняется некоторый код, а в любом месте программы можно написать `goto` и переместиться туда.

Примеры

- Определение совершеннолетия с помощью `if-else`

```c
int age;

printf("Введите ваш возраст: ");
scanf("%d", &age);

if (age >= 18) {
    printf("Вы совершеннолетний.\n");
} else {
    printf("Вы несовершеннолетний.\n");
}
```

- Нахождение максимального числа с использованием тернарного оператора

```c
int a, b, max;

printf("Введите два числа: ");
scanf("%d %d", &a, &b);

max = (a > b) ? a : b; // Выбираем большее из двух чисел
printf("Максимальное число: %d\n", max);
```

- Калькулятор действий с использованием `switch-case`

```c
int option;

printf("Выберите действие:\n");
printf("1. Сложение\n");
printf("2. Вычитание\n");
printf("3. Умножение\n");
printf("Введите ваш выбор (1-3): ");
scanf("%d", &option);

switch (option) {
    case 1:
        printf("Вы выбрали сложение.\n");
        break;
    case 2:
        printf("Вы выбрали вычитание.\n");
        break;
    case 3:
```

```
            printf("Вы выбрали умножение.\n");
            break;
        default:
            printf("Неверный выбор.\n");
            break;
    }
```