Conditionals

Conditionals

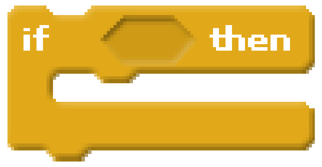
- Conditional expressions allow your programs to make decisions and take different forks in the road, depending on the values of variables or user input.

- C provides a few different ways to implement conditional expressions (also known as *branches*) in your programs, some of which likely look familiar from Scratch.

Conditionals

```
if (boolean-expression)
{

}
```



- If the `boolean-expression` evaluates to `true`, all lines of code between the curly braces will execute in order from top-to-bottom.

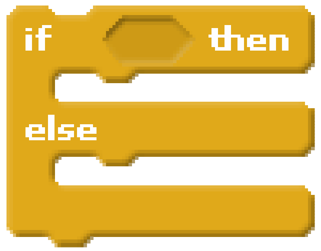
- If the `boolean-expression` evaluates to `false`, those lines of code will not execute.

Conditionals

```
if (boolean-expression)
{

}
else
{

}
```



- If the `boolean-expression` evaluates to `true`, all lines of code between the first set of curly braces will execute in order from top-to-bottom.

- If the `boolean-expression` evaluates to `false`, all lines of code between the second set of curly braces will execute in order from top-to-bottom.

Conditionals

```
if (boolean-expr1)
{
   // first branch
}
else if (boolean-expr2)
{
   // second branch
}
else if (boolean-expr3)
{
   // third branch
}
else
{
   // fourth branch
}
```

- In C, it is possible to create an `if–else if-else` chain.
  - In Scratch, this required nesting blocks.
- As you would expect, each branch is mutually exclusive.

Conditionals

```
if (boolean-expr1)
{
   // first branch
}
if (boolean-expr2)
{
   // second branch
}
if (boolean-expr3)
{
   // third branch
}
else
{
   // fourth branch
}
```

- It is also possible to create a chain of non-mutually exclusive branches.

- In this example, only the third and fourth branches are mutually exclusive. The `else` binds to the nearest `if` only.

Conditionals

```
int x = GetInt();
switch(x)
{
    case 1:
        printf(“One!\n”);
        break;
    case 2:
        printf(“Two!\n”);
        break;
    case 3:
        printf(“Three!\n”);
        break;
    default:
        printf(“Sorry!\n”);
}
```

- C’s `switch()` statement is a conditional statement that permits enumeration of discrete cases, instead of relying on Boolean expressions.

- It’s important to `break;` between each case, or you will “fall through” each case (unless that is desired behavior).

Conditionals

```
int x = GetInt();
switch(x)
{
   case 5:
      printf(“Five, ”);
   case 4:
      printf(“Four, ”);
   case 3:
      printf(“Three, ”);
   case 2:
      printf(“Two, ”);
   case 1:
      printf(“One, ”);
   default:
      printf(“Blast-
              off!\n”);
}
```

- C’s `switch()` statement is a conditional statement that permits enumeration of discrete cases, instead of relying on Boolean expressions.
- It’s important to `break;` between each case, or you will “fall through” each case (unless that is desired behavior).

Conditionals

```
int x;
if (expr)
{
    x = 5;
}
else
{
    x = 6;
}
```

```
int x = (expr) ? 5 : 6;
```

- These two snippets of code act identically.
- The ternary operator (? :) is mostly a cute trick, but is useful for writing trivially short conditional branches. Be familiar with it, but know that you won’t need to write it if you don’t want to.

Conditionals

`if` (and `if-else`, and `if-else if-…-else`)

- Use Boolean expressions to make decisions.

`switch`

- Use discrete cases to make decisions.

`?:`

- Use to replace a very simple `if-else` to make your code look fancy.