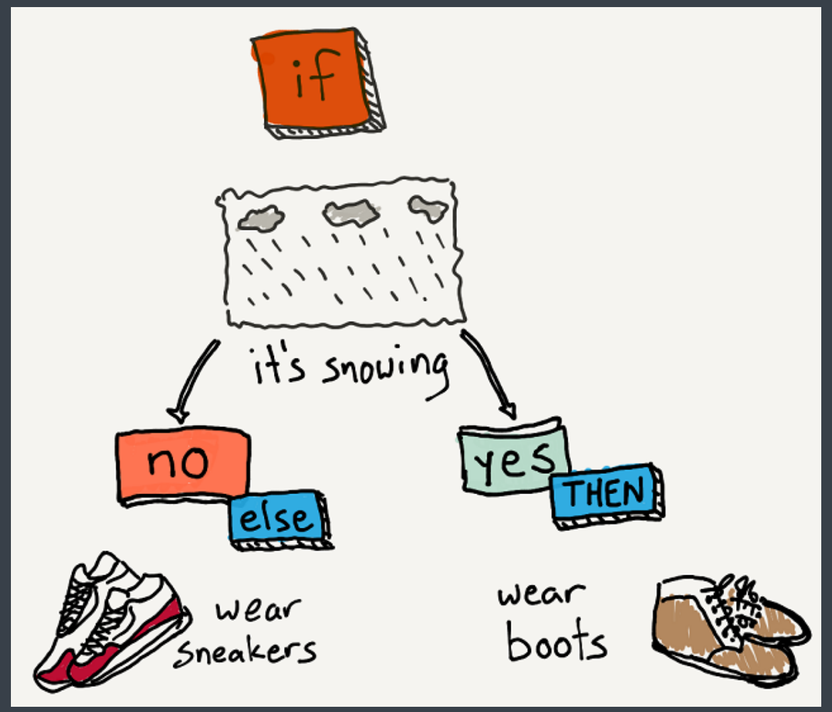
Decision Making

What is decision making ?

- The act or process of deciding something
- Based on certain criteria or conditions



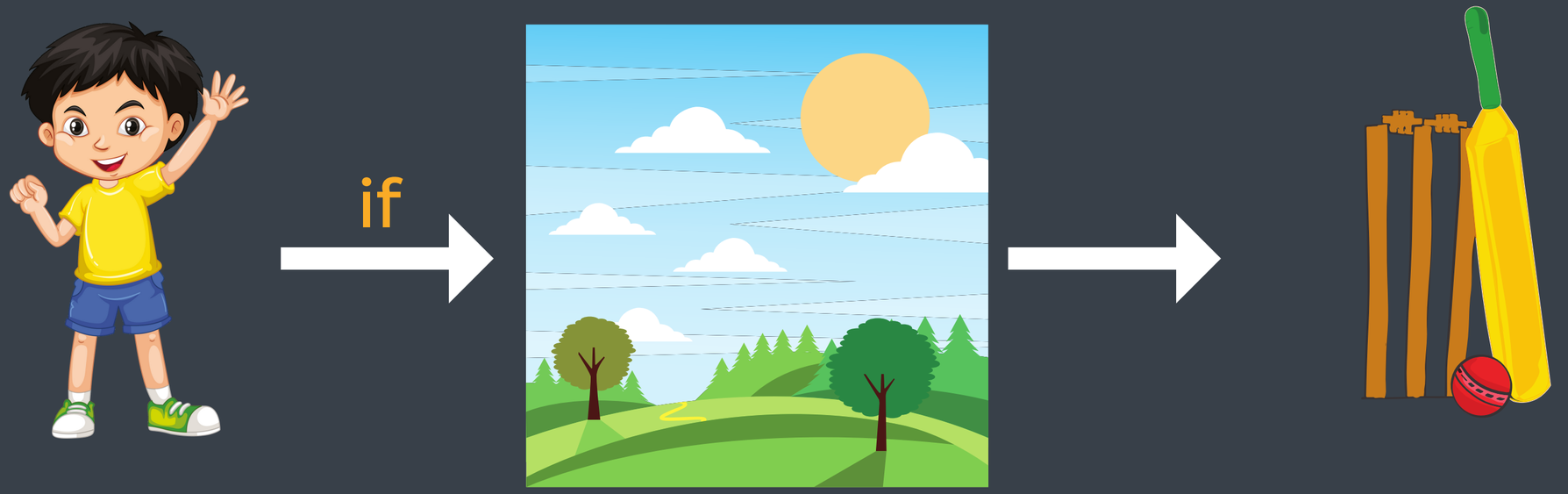
Conditional Statements

Conditional Statements

The most common conditional statements used for decision-making in programming are

- if
- if-else
- if-else if-else
- switch-case

If statement




```
if condition:
    # code to be executed if the condition is true
```

Syntax

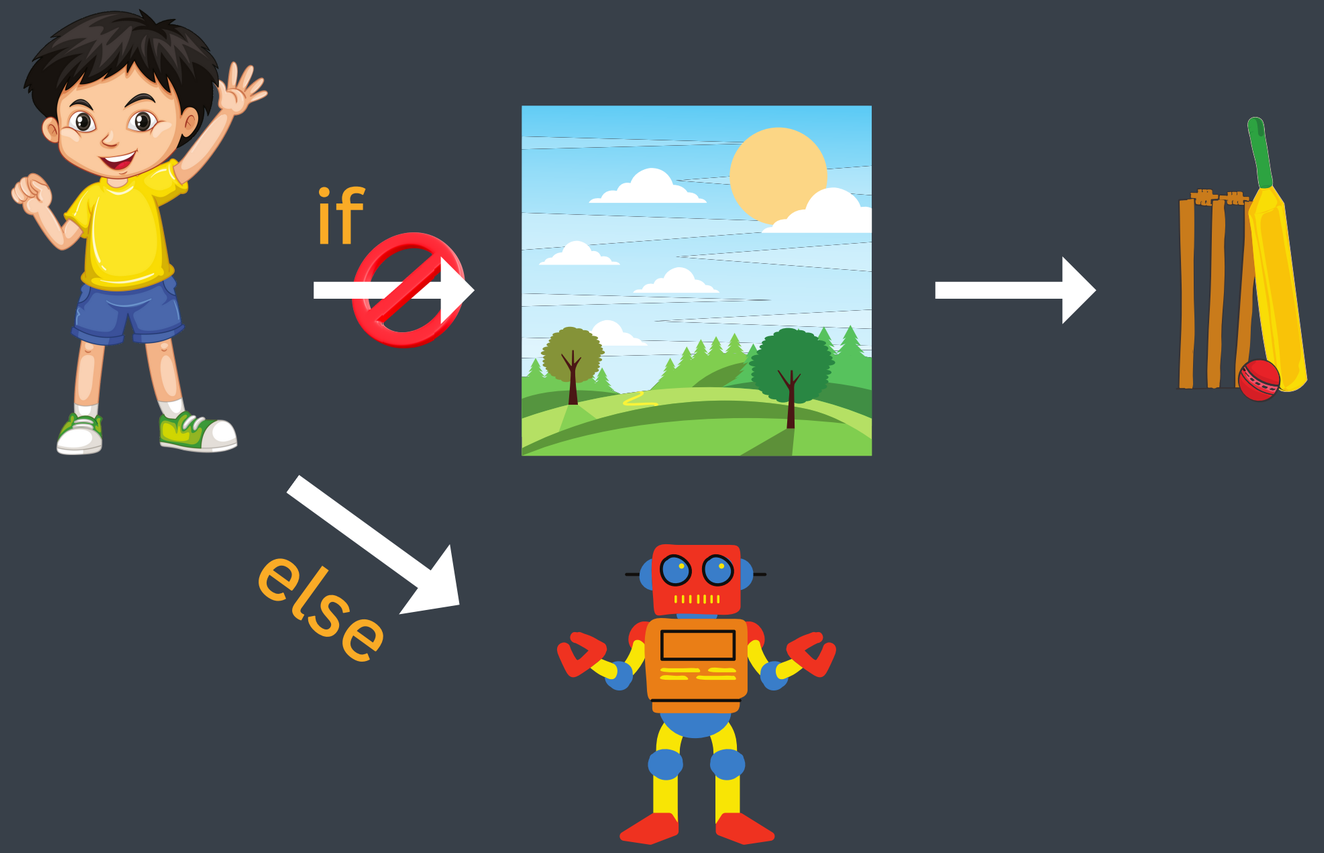
The condition is an expression that evaluates to either True or False. If the condition is true, the indented block of code under the if statement is executed; otherwise, it is skipped.

```
if (condition) {
    // Code to execute if the condition is true
}
```

Example

```c
#include <stdio.h>
int main() {
    int x = 5;
    if (x > 0) {
        printf("x is positive.\n");
    }
    return 0;
}
```

if-else Statement



Syntax

- If the condition is true, the code block under the if branch is executed, and the code block under the else branch is skipped.
- If the condition is false, the code block under the else branch is executed, and the code block under the if branch is skipped.

```
if condition{
   # code to be executed if the condition is true
}else{
   # code to be executed if the condition is false
}
```

Syntax

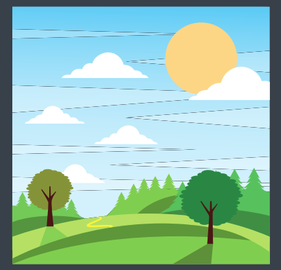
```
if (condition) {
    // Code to execute if the condition is true
} else {
    // Code to execute if the condition is false
}
```

Example

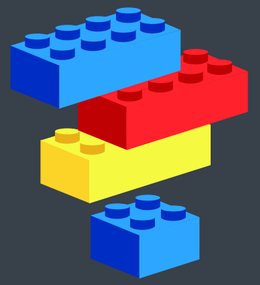
```c
#include <stdio.h>
int main() {
    int age = 17;
    if (age >= 18) {
        printf("You are eligible to vote.\n");
    } else {
        printf("You are not eligible to vote.\n");
    }
    return 0;
}
```

if-else if-else Statement

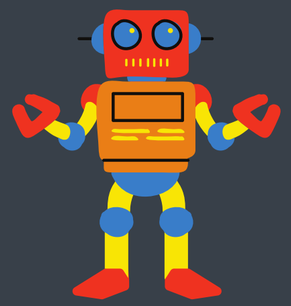
if



elif



else



```
if condition1{
    # code to be executed if condition1 is true
}else if condition2{
    # code to be executed if condition2 is true
}else ifcondition3{
    # code to be executed if condition3 is true
}else:
    # code to be executed if none of the conditions are true
}
```

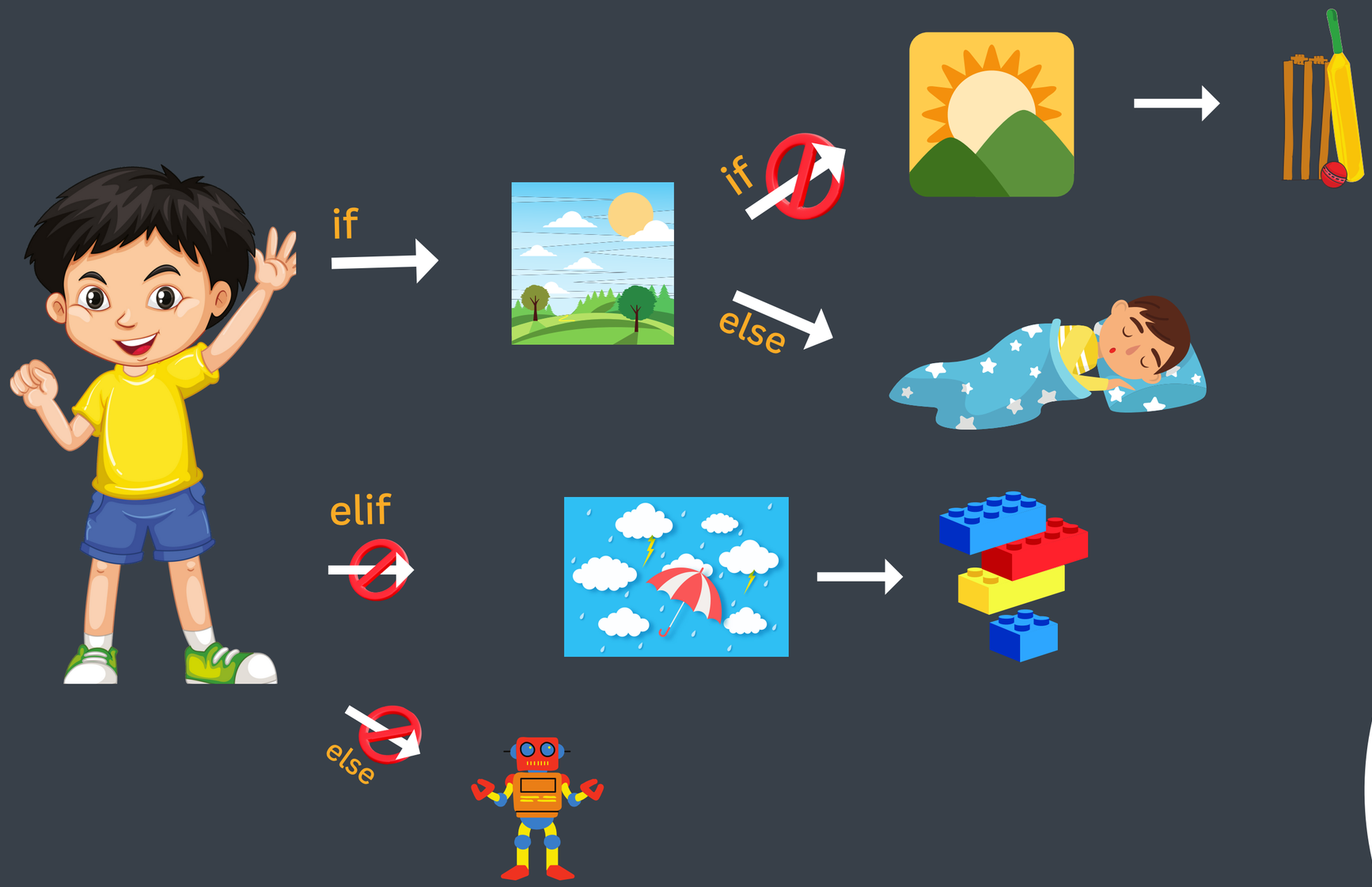
Syntax

```
if (condition1) {
    // Code to execute if condition1 is true
} else if (condition2) {
    // Code to execute if condition2 is true
} else {
    // Code to execute if no condition is true
}
```

Example

```c
#include <stdio.h>
int main() {
    int number = 7;
    if (number > 10) {
        printf("Number is greater than 10.\n");
    } else if (number < 10) {
        printf("Number is less than 10.\n");
    } else {
        printf("Number is equal to 10.\n");
    }
    return 0;
}
```

Nested if Statements



Syntax

```
if (condition1) {
    // Code to execute when condition1 is true

    if (condition2) {
        // Code to execute when both condition1 and condition2 are true
    } else {
        // Code to execute when condition1 is true, but condition2 is false
    }
} else {
    // Code to execute when condition1 is false
}
```

Example

```c
#include <stdio.h>
int main() {
    int age = 25;
    int income = 50000;
    if (age >= 18) {
        if (income >= 30000) {
            printf("You are eligible for a loan.\n");
        } else {
            printf("Your income is too low for a loan.\n");
        }
    } else {
        printf("You must be at least 18 years old to apply for a loan.\n");
    }
    return 0;
}
```

Logical operators

Logical operators in C are used to perform logical operations on boolean values (true or false). They allow you to combine multiple conditions and make decisions based on the truth values of those conditions.

- Logical AND (&&)
- Logical OR (||)
- Logical NOT (!)

Logical AND (&&)

The logical AND operator returns true if both of its operands are true.

```
if (condition1 && condition2) {
    // Code to execute if both conditions are true
}
```

Logical OR (||)

The logical OR operator returns true if at least one of its operands is true.

```
if (condition1 || condition2) {
    // Code to execute if at least one condition is true
}
```

Logical NOT (!)

The logical NOT operator negates the truth value of its operand. If the operand is true, ! makes it false, and vice versa.

```
if (!condition) {
    // Code to execute if the condition is false
}
```

Ternary operator

Is a shorthand way of writing an if-else statement. It allows you to conditionally assign a value to a variable based on a condition.

```
condition ? expression_if_true : expression_if_false;
```

- If the condition is true, the value of expression_if_true is returned.
- If the condition is false, the value of expression_if_false is returned.

Example

```c
#include <stdio.h>
int main() {
    int x = 5;
    int y = 10;
    int max = (x > y) ? x : y;
    printf("The maximum value is: %d\n", max);
    return 0;
}
```

Switch-Case

switch:

statement is used for decision-making by selecting one of many code blocks to be executed based on the value of an expression. It's a useful alternative to long chains of if-else if-else statements when you have a variable with discrete values to compare.

Syntax

```
switch (expression) {
    case value1:
        // Code to execute if expression matches value1
        break;
    case value2:
        // Code to execute if expression matches value2
        break;
    // Additional cases
    default:
        // Code to execute if no case matches
}
```

Example

```c
#include <stdio.h>
int main() {
    char grade = 'B';
    switch (grade) {
        case 'A': printf("Excellent!\n"); break;
        case 'B': printf("Good job!\n"); break;
        case 'C': printf("You passed.\n"); break;
        default:  printf("Please check your grade.\n");
    }
    return 0;
}
```

Simple Calculator Program (Project)

- Create a basic calculator program that performs addition, subtraction, multiplication, and division.
- Ask the user to enter two numbers and choose an operation.
- Display the result accordingly.
- Handle potential errors gracefully.