Change the flow of a program

Behram Khan

Lecture Overview

- In this first lecture we will cover the fundamental theoretical concepts that form the backbone of programming.
- This session focuses solely on theory along with practical examples, providing a solid foundation for understanding computers and programming
- Involving examples and interactive session to develop problem solving

Learning Objectives

By the end of this lecture, students will be able to:

- Know syntax of different statements
- Know how data is displayed on screen
- Know how data is taken from user as input
- Know how if statements work and nested if’s
- Know what are Boolean operators

Getch

Reads single character input from user but doesn’t wait for us to press enter key, as soon as we press a key, it displays the output.

Doesn’t show the pressed key on screen, Hide inputs like password, ATM pins…

Getch

```
Press any key to continue...
You pressed: a
```

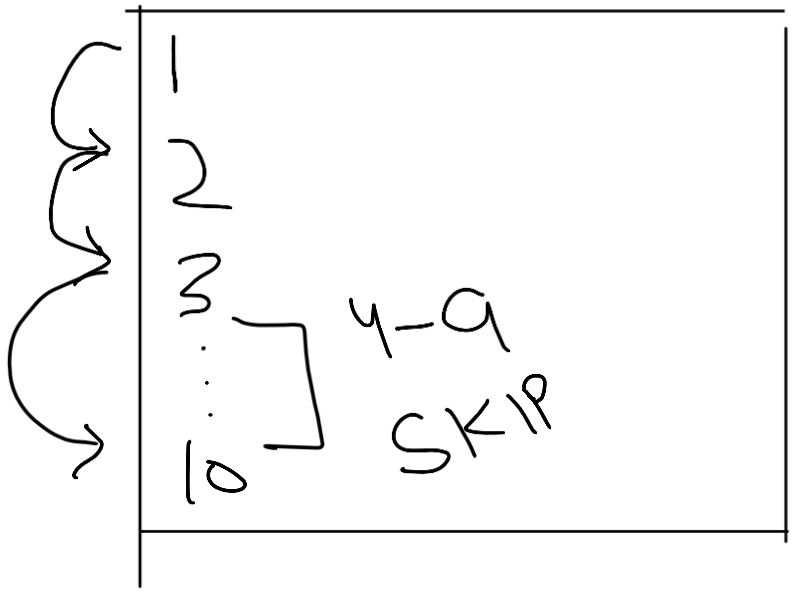
```c
#include <conio.h> // Library where getch() is stored
#include <stdio.h>
int main() {
    char ch;
    printf("Press any key to continue...");
    ch = getch(); // waits for a key press, doesn't display it
    printf("\nYou pressed: %c\n", ch);
    return 0;
}
```

Flow of Control

Flow of Control

Statements in a program execute one after the other in the order in which you write them. This is called **sequential execution.**

Various C statements enable you to specify that the next statement to execute may be other than the next one in sequence. This is called transfer of control.



Comparison Operators

These are operators used to compare values and evaluate to either true or false.
Values being compared have to be of same datatype

Operator	Description
==	Equal (two equal signs)
!=	Not equal
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

Expression	Result
5 == 5	True
5 != 5	False
5 > 5	False
5 < 5	False
5 >= 5	True
5 <= 5	True

Flow of Control

An algorithm is a step-by-step process for solving a problem

To write an algorithm we use Pseudo Code and Flowchart

Problem Statement 1

Turn the air conditioning on when the temperature is greater than or equal to 80 degrees or else turn it off.

Identify the Inputs, processing, outputs in the pseudocode.

Problem Statement 1 PseudoCode

if temperature >= 80

Turn AC on

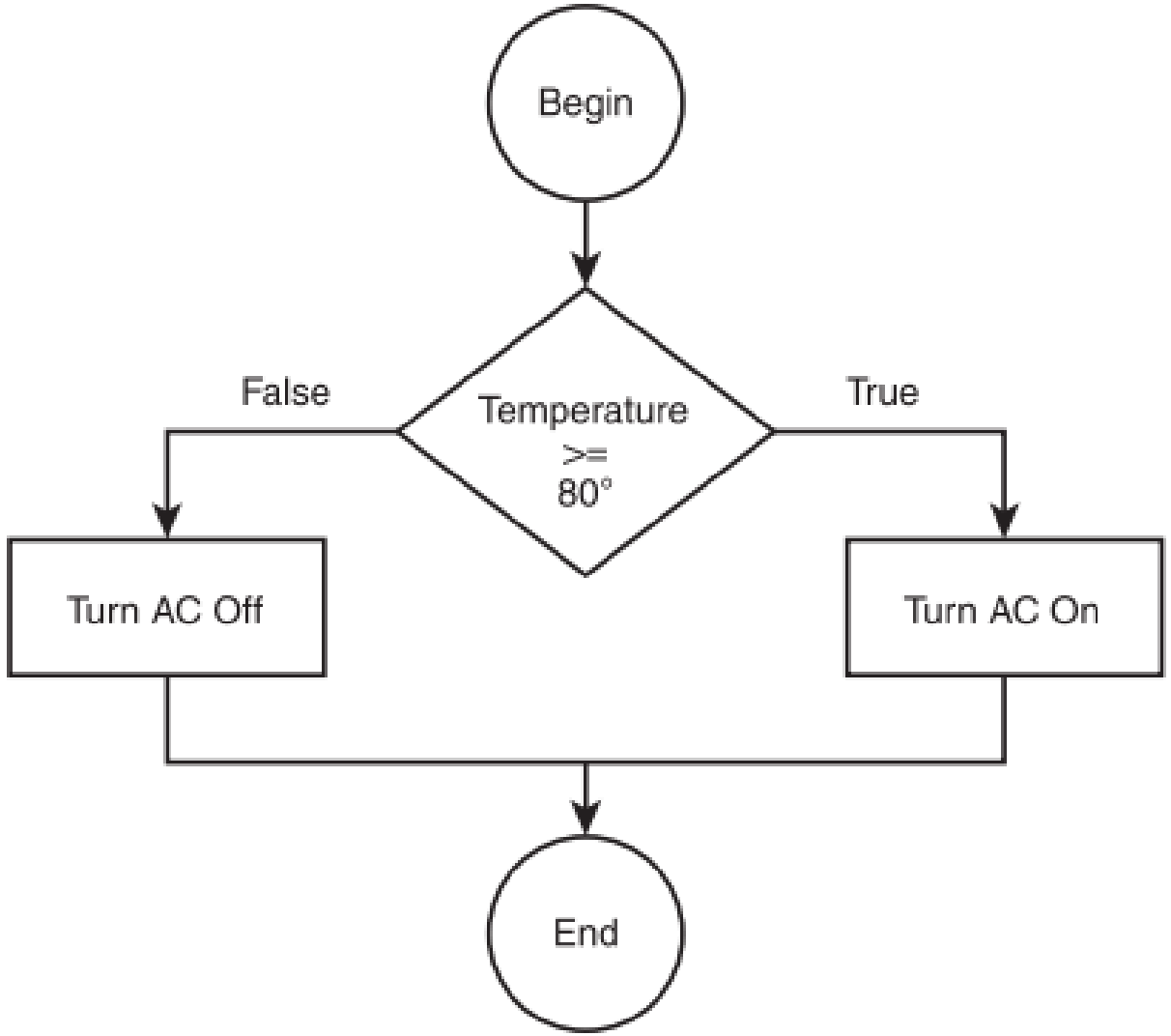
else

Turn AC off

end if

This is pseudocode not actual code, if u paste this in a C Program, it wont compile.

Problem Statement 1 Flowchart



Problem Statement 2

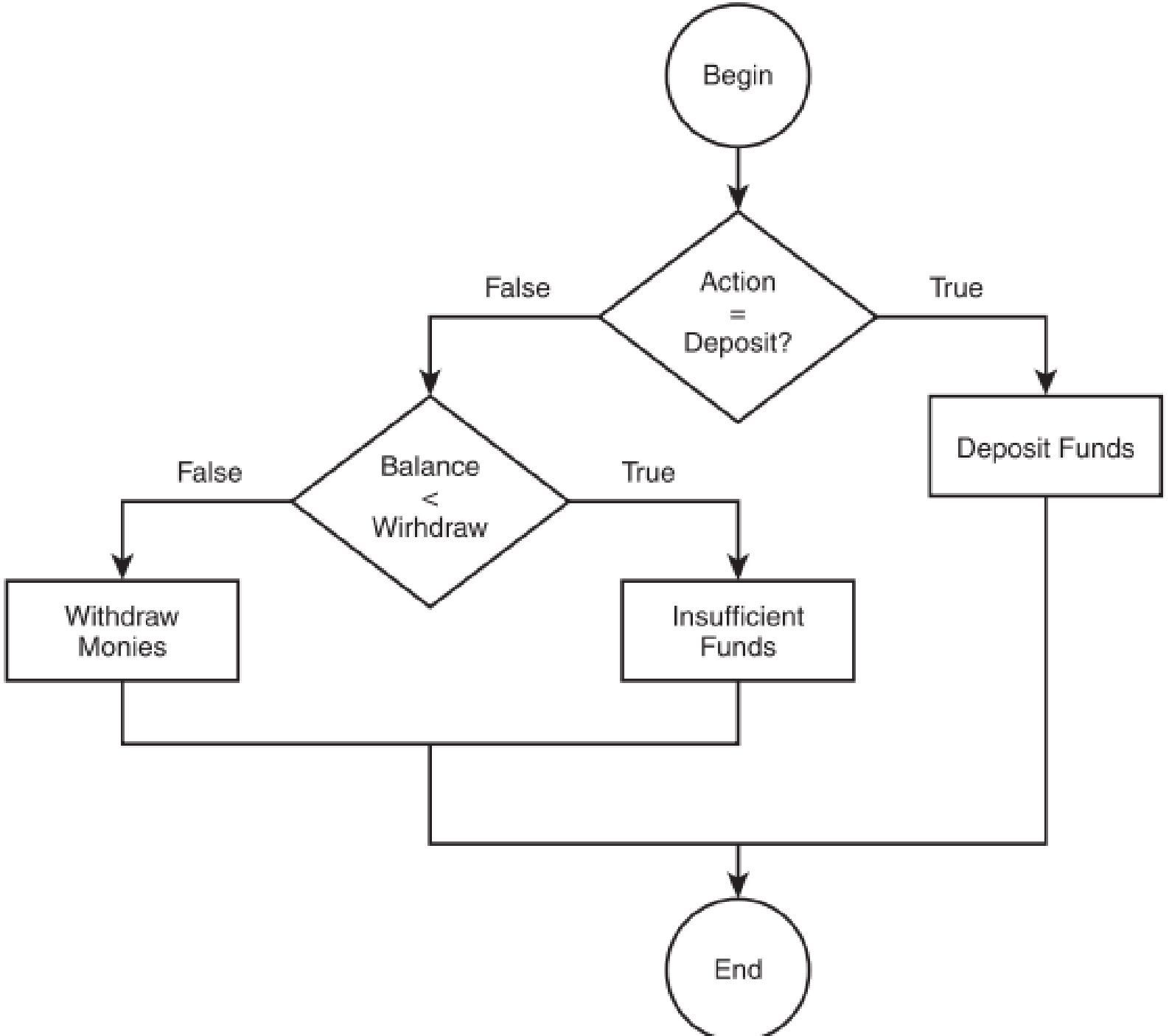
Allow a customer to deposit or withdraw money from a bank account, and if a user elects to withdraw funds, ensure that sufficient money exist.

Identify the Inputs, processing, outputs in the pseudocode.

Problem Statement 2 PseudoCode

```
if action == deposit
    Deposit funds into account
else
    if balance < withdraw amount
        Insufficient funds for transaction
    else
        Withdraw monies
    end if
end if
```

Problem Statement 2 Flowchart



If statement
A Decision-Making Tool

If Statement

An if statement allows you to execute a block of code only if a certain condition is true.

It's like a decision-making process: if a specific condition is met, you take one action; otherwise, you take another.

Syntax :

if (condition)

{ Code to be executed if the condition is true }

- **Condition:** This is an expression that evaluates to either true or false.
- **Code Block:** This is a block of code that will be executed if the condition is true.

Temperature Checking

```
if (Temperature >= 80) {
 //Turn AC on
 printf("\nThe AC is on\n");
}
If(Temperature <= 20)
 //Turn AC off
 printf("\nThe AC is off\n");
}
```

AcOnOff.c

Even Odd

```
if ( number % 2 == 0 ) {

 printf("\n Number is even \n");

}
```

EvenOdd.c

If else Statement

An if-else statement lets you execute one block of code if a specific condition is true, and a different block of code if that condition is false. It allows for two possible outcomes: one action when the condition is met and another when it isn't.

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

else  {    // Code to be executed if the condition is false}
```

- **Condition:** This is an expression that evaluates to either true or false.
- **If Block:** This block of code executes if the condition is true.
- **Else Block:** This block of code executes if the condition is false.

If else Statement

```
if (Temperature >= 80) {
 //Turn AC on
 printf("\nThe AC is on\n");
}
else {
 //Turn AC off
 printf("\nThe AC is off\n");
}
```

AcOnOff_If_Else.c

If else Statement

```
START
  SET number to 5
  IF number is greater than 0 THEN
     PRINT "The number is positive."
  ELSE
     PRINT "The number is not positive.“
End If
END
```

positiveNumber.c

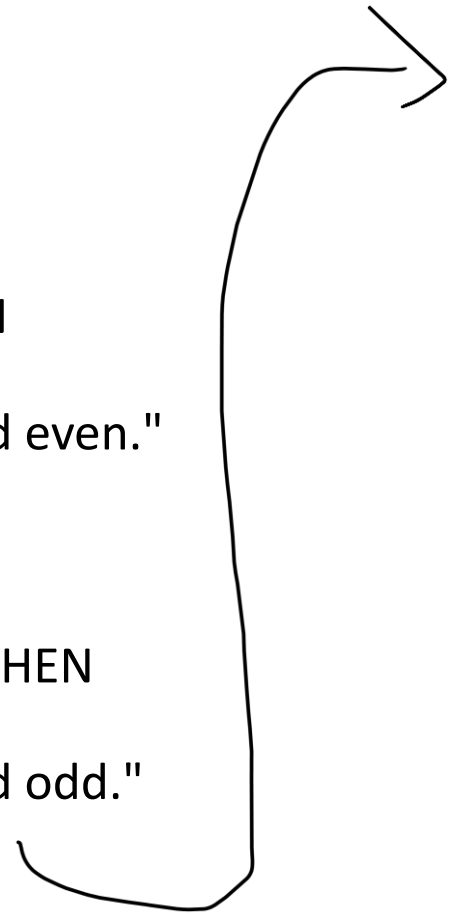
Nested If

A nested if statement is a decision-making structure that allows you to check multiple conditions within another if or else block.

In simple terms, it means having an if statement inside another if statement. This helps in making decisions based on more than one condition.

Checking if a number is positive, and if positive, whether it's even or odd.

```
START

  SET number to 8

  IF number is greater than 0 THEN

     IF number MOD 2 is equal to 0 THEN

        PRINT "The number is positive and even."

     END IF

     IF number MOD 2 is not equal to 0 THEN

        PRINT "The number is positive and odd."

                                                    END IF

                                                     END IF

                                                    END

                                                    PositiveAndEvenOdd.c
```



BehramKhan

Else If

Else if is a conditional statement that allows you to specify multiple conditions to be evaluated in a sequence.

- Adding condition to the else part of if else
- If no conditions are met, an optional ELSE can be used to handle all remaining cases.

Syntax:

if (condition1) { // Code to be executed if condition1 is true}

else if (condition2) { // Code to be executed if condition1 is false and condition2 is true}

else if (condition3) { // Code to be executed if condition1 and condition2 are false and condition3 is true}

else { // Code to be executed if none of the above conditions are true}

Else If

```
if ( number >  0 ) {

 printf("\n Number is Positive \n");

}

Else

{

Printf(“Number is Negative”);

}
```

Else If

```
#include <stdio.h>
int main()  {

        int number = 10;

        if (number > 0)
                { printf("The number is positive.\n"); }

        else if (number == 0)
                { printf("The number is zero.\n"); }

        else
                { printf("The number is negative.\n"); }

        return 0;
}
```

Problem Statement

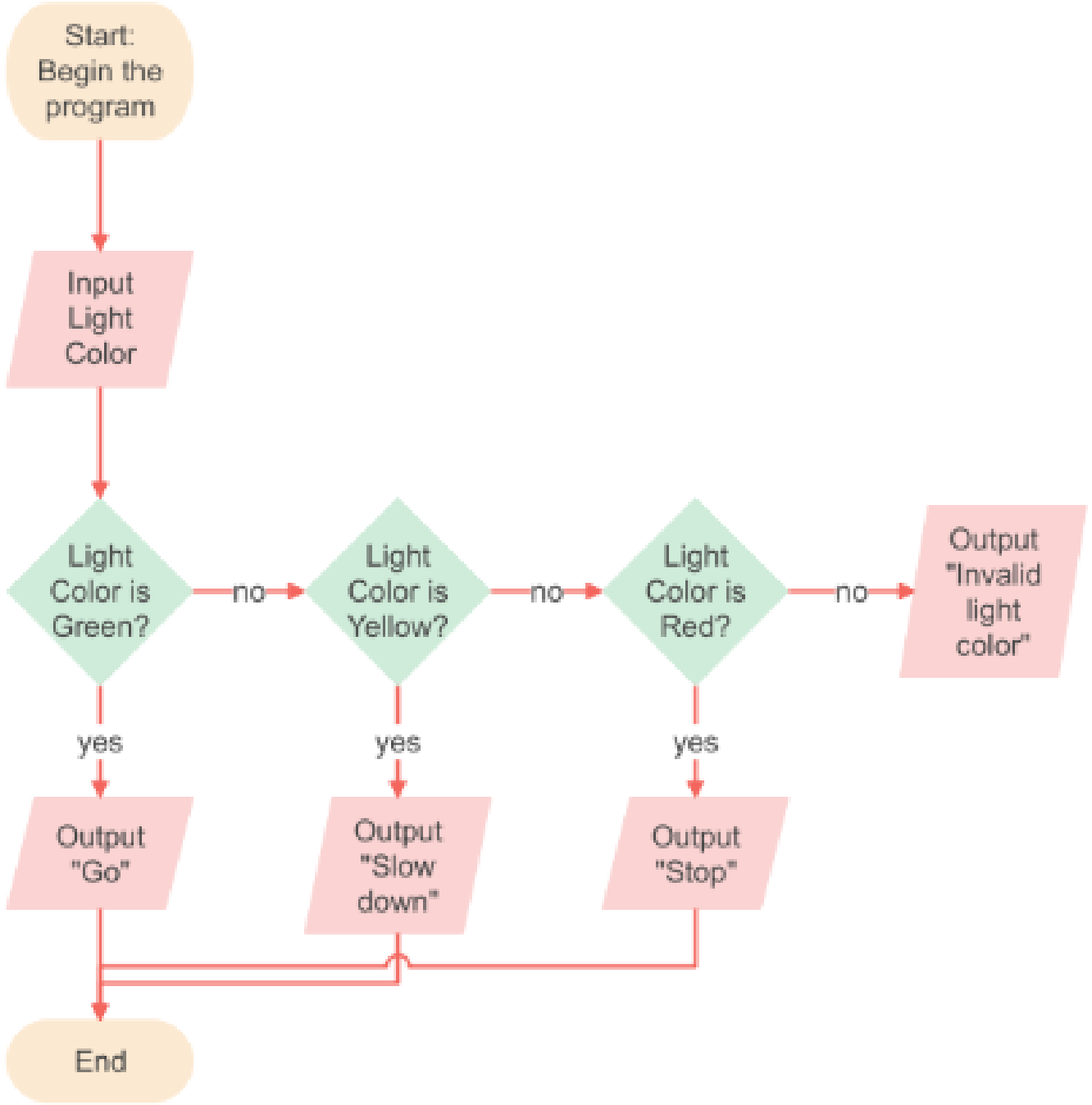
Create a program that simulates a traffic light system by displaying instructions based on the color of the traffic light. If the light is "green," the program should display "Go." If the light is "yellow," it should display "Slow down." If the light is "red," it should display "Stop." If the input does not match any of these colors, the program should display "Invalid light color.“

Identify Inputs, Outputs and Processing

Problem Statement

Input the color of the traffic light (stored in lightColor).
IF lightColor is equal to "green"
    THEN display "Go."
ELSE IF lightColor is equal to "yellow"
    THEN display "Slow down."
ELSE IF lightColor is equal to "red"
    THEN display "Stop."
ELSE
    Display "Invalid light color."
End IF

Flowchart


Start: Begin the program
Input Light Color
Light Color is Green?
no
Light Color is Yellow?
no
Light Color is Red?
no
Output "Invalid light color"
yes
yes
yes
Output "Go"
Output "Slow down"
Output "Stop"
End

Code

```
if (lightColor == "green")
{    printf("Go\n");  }

else if (lightColor == "yellow")
{    printf("Slow down\n");         }

else if (lightColor == "red")
{    printf("Stop\n");          }

else
{    printf("Invalid light color\n");                    }
```

Code

Last else of else if doesn’t have if statement. It gets executed only when all the previous statements have not run, meaning they were false.

Generally statements which have a body doesn’t need terminal ‘’ ; ’’ at end.

Boolean Algebra

Boolean Algebra

Boolean algebra is named after George Boole, a mathematician in the nineteenth century. Boole developed his own branch of logic containing the values true and false and the operators and, or, and not to manipulate the values.

- Boolean operators are used to combine or negate conditions in programming.

Three main Boolean operators are:

1. **AND**
2. **OR**
3. **NOT**

The AND Operator

The AND operator builds a compound conditions.

- Both conditions on either side must be true for the entire condition to be true
- **Example 1:**

3 == 3 AND 4 == 4

Result: True (Both conditions are true)

- **Example 2:**

3 == 4 AND 4 == 4

Result: False (Only one condition is true)

The AND Operator

x (Condition 1)	y (Condition 2)	Result (x AND y)
True	True	True
True	False	False
False	True	False
False	False	False

- **Explanation:** The AND operator only returns `true` when both sides are `true`.

The OR Operator

The OR operator builds a compound conditions.

- Only one condition needs to be true for the entire expression to be true.
- **Example 1:**

4 == 3 AND 4 == 4

Result: True (One conditions is true)

- **Example 2:**

3 == 4 AND 4 == 5

Result: False (Both conditions is false)

The OR Operator

x (Condition 1)	y (Condition 2)	Result (x OR y)
True	True	True
True	False	True
False	True	True
False	False	False

- **Explanation:** The OR operator returns `true` if at least one condition is `true`.

The NOT Operator

The NOT operator negates a condition.

- It inverts the value of a condition (if it's true, it becomes false; if it's false, it becomes true).
- **Example 1:**

NOT(4 == 4)

Result: False (4 == 4 is true, but NOT negates it)

x (Condition)	Result (NOT x)
True	False
False	True

The NOT Operator

x (Condition 1)	y (Condition 2)	Result (x OR y)
True	True	True
True	False	True
False	True	True
False	False	False

- **Explanation:** The OR operator returns `true` if at least one condition is `true`.

Combining Operators

AND, OR, and NOT operators can be combined to create more complex conditions.

- The order of operations is from Left to right :

1. NOT
2. AND
3. OR

- Use parentheses () to control the order in which conditions are evaluated.

Combining Operators

Example:

NOT(4 == 4) AND (5 == 5 OR 3 == 3)

First, evaluate the conditions inside the parentheses.

Then, apply the NOT operator.

Finally, apply the AND operator.

Combining Operators

Try to solve the following Boolean algebra problems, given

x = 5, y = 3, and z = 4

1. x > 3 and z == 4 → True

2. y >= 3 or z > 4 → True

3. NOT(x == 4 or y < z) → false

4. (z == 5 or x > 3) and (y == z or x < 10) → true

Boolean in C Programming

AND Operator (&&)

The AND (&&) operator checks if both conditions on either side are true.

- **Syntax:** condition1 && condition2
- **Result:** The entire expression is true only if both conditions are true.

Example 1:

- if (3 > 1 && 5 < 10)
- **Result:** True because both 3 > 1 and 5 < 10 are true.

Example 2:

- if (3 > 5 && 5 < 5)
- **Result:** False because 3 > 5 is false, so the entire condition fails.

OR Operator (||)

The OR (||) operator checks if at least one condition is true.

- **Syntax:** condition1 || condition2
- **Result:** The entire expression is true if either condition is true.

Example 1:

- if (3 > 5 || 5 <= 5)
- **Result:** True because 5 <= 5 is true (even though 3 > 5 is false).

Example 2:

if (3 > 5 || 6 < 5)

- **Result:** False because both conditions 3 > 5 and 6 < 5 are false.

NOT Operator (!)

The NOT (!) operator is used to invert a condition. It returns true if the condition is false, and false if the condition is true.

- **Syntax:** !condition
- **Result**: Reverses the truth value of the condition.

Example 1:

- if (!(3 > 5))
- **Result:** True because 3 > 5 is false, and the NOT operator inverts it to true.

Example 2:

- if (!(5 < 5))
- **Result:** True because 5 < 5 is false, and the NOT operator inverts it to true.

Combining AND (&&), OR (||), and NOT (!) Operators

```
if ( !( 3 > 5 && 4 == 4 ) || 7 == 7 )

    printf("Condition is true\n");
```

First, the && operator evaluates the left condition.

Then, the NOT operator negates the result.

Finally, the || operator checks if the final result is true or false.

Combining AND (&&), OR (||), and NOT (!) Operators

```
#include <stdio.h>


int main() {
  int a = 1;
  int b = 0;
  if ( a && b ) {
    printf("&& checks if both a and b are true so its not \n" );
  }
  if ( a || b ) {
  printf(" ||  checks if at least one of a or b is true so its true \n" );
  }
  if ( !(a && b) ) {


    printf("a && b is false, so !(a && b) becomes true.\n" );
  } }
```

Input Validation

Checking for a range of values is a common programming practice for input validation. You can use compound conditions and relational operators to check for value ranges, as shown in the following program:

InputValidation.c
VotingBoolean.c

Project

Days Until Birthday

Create a program that calculates and displays the number of days remaining until the user's next birthday. The program should prompt the user to enter Current Date (day and month) ,then prompt the user to enter their birth date (day and month) and then calculate the days between the current date and their upcoming birthday. If the birthday is today, it should display a congratulatory message instead. The program should consider leap years if applicable.

Main Menu

1. Welcome the user and tell about the app

2. Use Printf and \t \n

```
        ********************************************
        * Welcome to the Birthday Countdown App! *
        ********************************************

This app will help you find out how many days are left until your next birthday!
```

Taking Input

1. The program should prompt the user to enter Current Date (day and month) ,then prompt the user to enter their birth date (day and month)

2. Use Printf and Scanf, Will need some variables for Scanf

```
Enter the current month (1-12):
Enter the current day (1-31):
Enter your birth month (1-12):
Enter your birth day (1-31):
```

Validating Inputs

1. Validate the inputs
2. Current and Birth Days should be greater than 0 and less than or equal to 31
3. Current and Birth Months should be greater than 0 and less than or equal to 12
4. Use if with OR (||) to combine both conditions.

Days Left Calculation

1. Make a Variable to store the days remaining

2. **If the Birth Month Matches the Current Month**

If (BirthMonth == currentMonth)

if the user was born in March, it would calculate birthDay - currentDay

For example, if today were March 1 and their birthday was March 15, daysRemaining = 15 - 1 = 14.

Days Left Calculation

3. If the Birth Month is After the Current Month

If (birthMonth > currentMonth)

birthday is later in the year.

daysRemaining = (birthMonth - currentMonth) * 30 + (birthDay - currentDay);

Days Left Calculation

For example, if the current date is March 15 and the birthday is June 10:

birthMonth - currentMonth is 6 - 3 = 3 months away.

3 * 30 = 90 days (approximate days for 3 months).

Then, birthDay - currentDay gives 10 - 15 = -5 days.

The total is 90 + (-5) = 85 days.

Days Left Calculation

4. If the Birth Month is Before the Current Month

If birthMonth < currentMonth

daysRemaining = (12 - currentMonth + birthMonth) * 30 + (birthDay - currentDay);

For example, if today is November 15 (month 11) and the birthday is March 10:

12 - currentMonth + birthMonth is 12 - 11 + 3 = 4 months away.

4 * 30 = 120 days.

birthDay - currentDay gives 10 - 15 = -5 days.

Total is 120 + (-5) = 115 days.

Displaying Birthday based on days remaining

Birthday Today
if(daysRemaining == 0): If daysRemaining is exactly zero, print "Happy Birthday!".

Birthday Within 30 Days
else if (daysRemaining < 30 && daysRemaining > 0):
For birthdays within the next month, print "Your birthday is just around the corner!".

Birthday in a Few Months
else if (daysRemaining >= 30 && daysRemaining <= 180):
For birthdays between 1 month and 6 months away, print "Your birthday is a few months away.".

Birthday in Almost a Year
else: If the birthday is more than 6 months away, print "Your birthday is in almost a year!".

THANK YOU!