**Programming Example**

**The Game of Rock, Paper, and Scissors**

Children often play the game of rock, paper, and scissors. This game has two players, each of whom chooses one of the three objects: rock, paper, or scissors. If player 1 chooses rock and player 2 chooses paper, player 2 wins the game because paper covers the rock. The game is played according to the following rules:

* If both players choose the same object, this play is a tie.
* If one player chooses rock and the other chooses scissors, the player choosing the rock wins this play because the rock breaks the scissors.
* If one player chooses rock and the other chooses paper, the player choosing the paper wins this play because the paper covers the rock.
* If one player chooses scissors and the other chooses paper, the player choosing the scissors wins this play because the scissors cut the paper.

Write an interactive program that allows two people to play this game.

|  |  |
| --- | --- |
| **Input** | This program has two types of input: |
|  | • The users’ responses when asked to play the game. |
|  | • The players’ choices. |
| **Output** | The players’ choices and the winner of each play. After the game is over, the total number of plays and the number of times that each player won should be output as well. |

**Problem Analysis and Algorithm Design**

Two players play this game. Players enter their choices via the keyboard. Each player enters R or r for Rock, P or p for Paper, or S or s for Scissors. While the first player enters a choice, the second player looks elsewhere. Once both entries are in, if the entries are valid, the program outputs the players’ choices and declares the winner of the play. The game continues until one of the players decides to quit the game. After the game ends, the program outputs the total number of plays and the number of times that each player won. This discussion translates into the following algorithm:

1. Provide a brief explanation of the game and how it is played.
2. Ask the users if they want to play the game.
3. Get plays for both players.
4. If the plays are valid, output the plays and the winner.
5. Update the total game count and winner count.
6. Repeat Steps 2 through 5 while the users agree to play the game.
7. Output the number of plays and times that each player won.

We will use the following enumeration type to describe the objects.

**Variables (Function main)**

It is clear that you need the following variables in the function main:

This program is divided into the following functions, which the ensuing sections describe in detail.

* **displayRules:** This function displays some brief information about the game and its rules.
* **validSelection:** This function checks whether a player’s selection is valid. The only valid selections are R, r, P, p, S, and s.
* **retrievePlay:** Because enumeration types cannot be read directly, this function converts the entered choice (R, r, P, p, S, or s) and returns the appropriate object type.
* **gameResult:** This function outputs the players’ choices and the winner of the game.
* **convertEnum:** This function is called by the function gameResult to output the enumeration type values.
* **winningObject:** This function determines and returns the winning object.
* **displayResults:** After the game is over, this function displays the final results.

**Function displayRules**

This function has no parameters. It consists only of output statements to explain the game and rules of play. Essentially, this function’s definition is:

**Function validSelection**

This function checks whether a player’s selection is valid.

Let’s use a switch statement to check for the valid selection. The definition of this function is:

**Function retrievePlay**

Because the enumeration type cannot be read directly, this function converts the entered choice (R, r, P, p, S, or s) and returns the appropriate object type. This function thus has one parameter, of type char. It is a value-returning function, and it returns a value of type objectType. In pseudocode, the algorithm of this function is:

The definition of the function retrievePlay is:

**Function gameResult**

This function decides whether a game is a tie or which player is the winner. It outputs the players’ selections and the winner of the game. Clearly, this function has three parameters: player 1’s choice, player 2’s choice, and a parameter to return the winner. In pseudocode, this function is:

1. if player1 and player2 have the same selection, then this is a tie game.

The definition of this function is:

**Function convertEnum**

Because enumeration types cannot be output directly, let’s write the function convertEnum to output objects of the enum type objectType. This function has one parameter, of type objectType. It outputs the string that corresponds to the objectType. In pseudocode, this function is:

The definition of the function convertEnum is:

**Function winningObject**

To decide the winner of the game, you look at the players’ selections and then at the rules of the game. For example, if one player chooses ROCK and another chooses PAPER, the player who chose PAPER wins. In other words, the winning object is PAPER. The function winningObject, given two objects, decides and returns the winning object. Clearly, this function has two parameters of type objectType, and the value returned by this function is also of type objectType. The definition of this function is:

**Function displayResults**

After the game is over, this function outputs the final results—that is, the total number of plays and the number of plays won by each player. The total number of plays is stored in the variable gameCount, the number of plays won by player 1 is stored in the variable winCount1, and the number of plays won by player 2 is stored in the variable winCount2. This function has three parameters corresponding to these three variables. Essentially, the definition of this function is:

We are now ready to write the algorithm for the function main.

**Main Algorithm**

1. Declare the variables.
2. Initialize the variables.
3. Display the rules.
4. Prompt the users to play the game.
5. Get the users’ responses to play the game.
6. while (response is yes)

{

* 1. Prompt player 1 to make a selection.
  2. Get the play for player 1.
  3. Prompt player 2 to make a selection.
  4. Get the play for player 2.
  5. If both plays are legal:

{

* + 1. Increment the total game count.
    2. Declare the winner of the game.
    3. Increment the winner’s game win count by 1.

}

* 1. Prompt the users to determine whether they want to play again.
  2. Get the players’ responses.

}

1. Output the game results.