Table of Contents

[Project Overview 2](#_Toc67217044)

[Top Level Functions 3](#_Toc67217045)

[ResultT (Enumeration) 5](#_Toc67217046)

[PlayerT (Structure) 6](#_Toc67217047)

# Project Overview

This project is to implement an interactive text based version of the game Hi-Ho Cherry-O! This game is played with four players, either human or computer. The implementation will involve enumerated types, structures and an array of players.

**Structure Chart**



# Top Level Functions

The main logic simulates the playing of a game.

This program, will use an array of players.

**Main**

Narrative: For each round, allow players to take a turn until a winner is found.

Get Players Info

While not done with the game

 For each player

 Take a turn

 If the player has collected all ten cherries

 Finished is true

 Print the current status of the game

Print the winning statement

**Function:** GetPlayersInfo

Narrative:

Input: An uninitialized array of players.

Output: The array of players with all players initialized.

For each player

 Set computer players to 0

 GetPlayerInformation(player, computerPlayers)

**Function:** GetPlayerInformation

Narrative:This will get the information for an individual player

Input: The empty player structure and the number of computer players

Output: The completed player structure

GetPlayer using the player function

If the player is a computer, increment the computer players by 1.

**Function:** PlayerTakeTurn

Narrative: This function will allow a player to take a trun.

Input: The player

Output: The modified player

Spin the spinner

Compute the number of cherries to move based on the player’s bucket and the result of the spinner

Move the correct number of cherries.

**Function:** ComputeCherriesToMove

Narrative: Compute the number of cherries to move based on the spinner and the bucket.

Input: A spinner result, the number of cherries in the plyer’s bucket.

Output:The number of cherries to move.

Move = GetRawResultMove

If the spiner is spill bucket

 Move = bucket

If bucket is less than 2 and move is -2

 Move = bucket

If bucket + move > 10

 Move = 0

Return move

**Function:** GetRawCherriesMove

Narrative: Compute the raw result move

Input: A spinner result,

Output:The number of cherries to move.

This is a big case statement.

If the result is bucket,

 return -10

If the result is bird or dog
 return -2

Otherwise return the number of cherries.

# ResultT (Enumeration)

ResultT will describe the outcome of a spin on the spinner.

Properties

A ResultT may be

ONE\_CHERRY BIRD

TWO\_CHERRIES DOG

THREE\_CHERRIES SPILL\_BASKET

FOUR\_CHERRIES UNKNOWN

Behaviors

 Convert a ResultT to a string.

 Convert an integer to a ResultT

 Generate a random ResulT

Special Considerations

 Create a constant, NUMBER\_OF\_RESULTS that represents the number of possible results.

**Function:** ResultTToString

Narrative: Provide a string for the given ResultT

Input: A ResultT

Output: A string

Just a large switch/case statement that assigns the correct string to a resultT

**Function:** IntToResultT

Narrative: convert an integer to a ResultT

Input: an integer.

Output: a ResultT

Cast the given integer to a result.

This will map the integer into the proper range using the mod function.

**Function:** RandomResulT

Narrative: This will generate a random result uniformly distributed between all values except UNKNOWN which will not be included.

Input: None

Output: A random result.

Generate a random integer between 0 and NUMBER\_OF\_RESULTS-1

Return the value of this using IntToResulT

# PlayerT (Structure)

This structure represents a player in the game.

Properties

 The player’s name. A simple string

 The player’s type (Human or Computer)

 The count of cherries in the player’s bucket, an integer.

Behaviors

 Print a player

 Get Player

 Get a human player

 Generate a computer player

**Function:** Print Player

Narrative: This will print information about a player

Input: The player structure

Output: None

Print the player name, type and bucket count.

**Function:**GetPlayer

Narrative: This function will ask for details about a human player.

Input: The number of computer players so far

Output: A completed player structure

 A modified number of computer players

Set bucket to 0

Ask for type

If type is human

 GetHumanPlayer

Else

 GetComputerPlayer

**Function:**GetHumanPlayer

Narrative: This function will ask for details about a human player.

Input: A partially completed player structure

Output: The completed structure

Get the human player name from the player

Set the type to be human

**Function:**GetComputerPlayer

Narrative: This function will set the details for a computer player

Input: A partially completed player structure

 The current number of computer players

Output: The completed structure

Get the name based on the number of computer players so far

Set the type to be computer.