﻿using System;

using System.Collections.Generic;

using System.Linq;

namespace AmigoInvisible

{

static class Program

{

//Practica 1 amigo invisible

//Hecha por Andrés Ragot, José Salvatierra y Alicia Touris

//este array de las parejas lo inicializamos aca ya que las parejas son un caso particular y en todo momento debemos compararlas y tener este array nos ayuda a esto, tambien nos asegura que cada vez que se introduzca una nueva pareja no se reinicialice

static string[] parejas = new string[30];

//funcion que hace el randomizado de los regalados y los regaladores

static void Shuffle(Random rng, string[] array)

{

//este metodo funciona de que agarramos el ultimo valor y lo randomizamos, para que el random no se pase y aseurarnos de que los ultimos se randomicen

int n = array.Length;

while(n>1)

{

int k = rng.Next(n--);

string temp = array[n];

array[n] = array[k];

array[k] = temp;

}

}

//esta funcion nos permite de hacer el randomizado con la misma logica pero de una manera mas organizada ya que con el circular nos aseguramos que un valor ya puesto en nuestro array de random se elimine

static string[] ShuffleCircula(Random rng, string[] array2)

{

//usamos las listas ya que de esta manera podemos borrar de manera rapida los elementos que ya usamos y llenamos otra de la manera deseada

List<string> temp = new List<string>(array2);

List<string> temp2 = new List<string>();

int n = array2.Length;

while (temp.Count()!=0)

{

int i = 0;

int k = rng.Next(n--);

temp2.Add(temp[k]);

temp.Remove(temp[k]);

i++;

}

array2 = temp2.ToArray();

return array2;

}

// esta funcion nos permite comparar si alguien se regala a si mismo

static bool Comparar(string[] array, string[] array2)

{

bool resultado = true;

//verificamos que no esten vacios aunque para poder llamar esta funcion ya antes paso por esta verificacion. esto nos ayudo mas que todo para las partes de preparacion antes de hacer el menu

if (array != null && array2 != null)

{

//recorremos el array y comparamos el valor en la misma linea para saber si es el mismo o no

for (int i = 0; i < array2.Length; i++)

{

if (array[i].Equals(array2[i]))

{

//regresamos falso ya que no se hizo bien el shuffle

resultado = false;

}

}

}

//si estan vacios nos dice que estan vacios

else

{

Console.WriteLine("Problema los arrays proporcionados están vacios");

}

return resultado;

}

//esta funcion compara si las parejas se estan regalando entre ellas

static bool CompararParejas(string[] array, string [] array2)

{

//utilizamos el comparar de antes ya que no vamos a volver a escribir codigo ya hecho

bool resultado = Comparar(array, array2);

if (array!=null && array2 != null)

{

for(int i = 0; i < array2.Length; i++)

{

//recorremos ambos arrays

for(int j = 0; j<Length(parejas); j++)

{

//comparamos si uno de los regaladores forma parte de una pareja

if (array[i].Equals(parejas[j]))

{

//y verficamos la posicion de la pareja. si es divisible entre 2 sgnifica que fue la primera persona que se introdujo al dar las parejas y significa que su pareja esta una posicion mas que la de el

if (j % 2 == 0)

{

if (array2[i].Equals(parejas[j + 1]))

{

resultado = false;

}

}

// si no es divisible entre 2 significa que esta es la segunda paraja administrada, entonces su pareja se encuentra una posicion menor

else if(array2[i].Equals(parejas[j - 1]))

{

resultado = false;

}

}

}

}

}

return resultado;

}

//esta funcion compara a ver si hacemos el shuffle de manera circular

static bool CompararCircular(string[] array, string[] array2)

{

//utilizamos el comparar parejas

int cantidadPersonasEnCirculo = 0;

bool resultado = CompararParejas(array,array2);

if (array != null && array2 != null)

{

//verificamos que empecemos por el primer regalado

int siguiente = Array.IndexOf(array, array2[0]);

//seguimos en este bucle con tal de no volver a encontrar al primer regalador

while (!array[0].Equals(array2[siguiente]))

{

//por cada vez que entremos identificamos a quien regala el regalador y aumentamos la cantidad de personas que tenemos

siguiente = Array.IndexOf(array, array2[siguiente]);

cantidadPersonasEnCirculo++;

}

//verificamos de esta manera, ya que como al entrar no cuenta ni al salir obtendremos, si se ha hecho correctamente la distancia del array menos dos

if (cantidadPersonasEnCirculo != array2.Length - 2)

{

//devolvemos falso si no se cumple

resultado = false;

}

}

return resultado;

}

//esta funcion nos permite imprimir los array de manera facil y rapida

static void ToString(string[] vs)

{

//por cada valor dentro de nuestro array

foreach (string value in vs)

{

//verificamos que los valores no sean nulos

if (value != null)

{

Console.WriteLine("\t\t\t\t"+value);

}

}

}

//esta manera nos permite imprimir de manera organizada dos arrays

static void ToString(string[] array, string[] array2)

{

int i = 0;

//aca utiizamos el array2 porque se inicializa a partir del array que es muy grande pero el array2 es justo lo que necesitamos

foreach(var value in array2)

{

Console.WriteLine("\t\t\t\t"+array[i] + "\t-->\t" + array2[i]);

i++;

}

}

//este menu nos ayuda a imprimir el menu del modo basico

static void MenuBasico()

{

Console.WriteLine("\t\t\t\t1 - Introducir nombre\n\t\t\t\t2 - ¡Amigo invisible!\n\t\t\t\t3 - Salir");

}

//este menu nos ayuda a imprimir el menu del modo parejas

static void MenuPareja()

{

Console.WriteLine("\t\t\t\t1 - Introducir nombre\n\t\t\t\t2 - Introducir pareja\n\t\t\t\t3 - ¡Amigo invisible!\n\t\t\t\t4 - Salir");

}

//este menu nos ayuda a imprimir el menu del modo circular

static void MenuCircular()

{

Console.WriteLine("\t\t\t\t1 - Introducir nombre\n\t\t\t\t2 - Introducir pareja\n\t\t\t\t3 - ¡Amigo invisible basico!\n\t\t\t\t4 - ¡Amigo invisible avanzado!\n\t\t\t\t5 - Salir");

}

//este menu nos ayuda a imprimir el menu del modo principal

static void MenuPrincipal()

{

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("\t\t\t\t\* \* \* \* \* \* \* \* \* \* \* \*\n\t\t\t\t\* \* \* Andrés Ragot \* \*\n\t\t\t\t\* \* José Salvatierra \*\n\t\t\t\t\* \* Alicia Touris \* \* \*\n\t\t\t\t\* \* \* \* \* \* \* \* \* \* \* \*");

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("\n\t\t\t\tEscoja un Modo:");

Console.WriteLine("\n\t\t\t\t1 - Modo Basico\n\t\t\t\t2 - Modo Pareja\n\t\t\t\t3 - Modo Circular/Avanzado");

}

//esta funcion nos ayuda a tener el valor exacto de la longitud del array. ya que los inicializamos con el limite que podemos tener, muchos valores seran nulos y no nos vale para lo que queremos

static int Length(string[] array)

{

int i = 0;

foreach(string value in array)

{

if (value != null)

{

i++;

}

}

return i;

}

//esta funcion nos ayuda a introducir el nombre de manera facil

static string[] IntroducirNombre(string[] array)

{

int i = Length(array);

array[i] = Console.ReadLine();

//imprimimos las peronas que llevamos agregadas

Console.WriteLine("\n\t\t\t\tPersonas agregadas hasta el momento\n");

ToString(array);

return array;

}

//esta funcion nos ayuda a introducir el nombre de manera facil y empezar el randomizado de manera automatica

static string[] IntroducirNombreRandomizado(string[] array, string[] array2)

{

Random random = new Random();

int i = Length(array);

array[i] = Console.ReadLine();

i = Length(array);

//le damos una linea mas para que cuando se imprima no moleste a los ojos

Console.WriteLine();

array2 = new string[i];

Array.Copy(array, array2, i);

//hacemos el shuffle hasta que el comparar nos de un true

do

{

Shuffle(random, array2);

} while (!Comparar(array, array2));

ToString(array, array2);

//regresamos el array porque tenemos que guardar el valor que acabamos de introducir

return array;

}

//esta funcion nos ayuda a introducir el nombre de manera facil y empezar el randomizado de manera automatica utilizando el random parejas

static string[] IntroducirNombreRandomizadoPareja(string[] array, string[] array2)

{

//lo mismo que el anterior pero comparamos con parejas y no con el basico

Random random = new Random();

int i = Length(array);

array[i] = Console.ReadLine();

i = Length(array);

Console.WriteLine();

array2 = new string[i];

Array.Copy(array, array2, i);

do

{

Shuffle(random, array2);

} while (!CompararParejas(array, array2));

ToString(array, array2);

return array;

}

//esta funcion nos ayuda a introducir el nombre de manera facil y empezar el randomizado de manera automatica utilizando el random circular

static string[] IntroducirNombreRandomizadoCircular(string[] array, string[] array2)

{

Random random = new Random();

int i = Length(array);

array[i] = Console.ReadLine();

i = Length(array);

Console.WriteLine();

array2 = new string[i];

Array.Copy(array, array2, i);

do

{

Shuffle(random, array2);

} while (!CompararCircular(array, array2));

ToString(array, array2);

return array;

}

//esta funcion nos ayuda a introducir los nombres de las parejas de manera facil

static string[] IntroducirPareja(string[] array)

{

int i = Length(array);

int j = Length(parejas);

Console.WriteLine("\t\t\t\tNombre de la primera persona");

array[i] = Console.ReadLine();

//las agregamos tambien a parejas para tenerlas apartadas

parejas[j] = array[i];

Console.WriteLine("\t\t\t\tNombre de la segunda persona");

array[i + 1] = Console.ReadLine();

parejas[j+1] = array[i+1];

Console.WriteLine();

Console.WriteLine("\t\t\t\tPersonas agregadas hasta el momento");

Console.WriteLine();

ToString(array);

return array;

}

//esta funcion nos ayuda a introducir los nombres de las parejas de manera facil y empezar el randomizado de manera automatica

static string[] IntroducirParejaRandomizado(string[] array, string[] array2)

{

Random random = new Random();

int i = Length(array);

int j = Length(parejas);

Console.WriteLine("\t\t\t\tNombre de la primera persona");

array[i] = Console.ReadLine();

parejas[j] = array[i];

Console.WriteLine("\t\t\t\tNombre de la segunda persona");

array[i + 1] = Console.ReadLine();

parejas[j + 1] = array[i + 1];

i = Length(array);

array2 = new string[i];

Array.Copy(array, array2, i);

do

{

Shuffle(random, array2);

} while (!CompararParejas(array, array2));

ToString(array, array2);

return array;

}

//esta funcion nos ayuda a introducir los nombres de las parejas de manera facil y empezar el randomizado de manera automatica utliziando el random circular

static string[] IntroducirParejaRandomizadoCircular(string[] array, string[] array2)

{

Random random = new Random();

int i = Length(array);

int j = Length(parejas);

Console.WriteLine("\t\t\t\tNombre de la primera persona");

array[i] = Console.ReadLine();

parejas[j] = array[i];

Console.WriteLine("\t\t\t\tNombre de la segunda persona");

array[i + 1] = Console.ReadLine();

parejas[j + 1] = array[i + 1];

i = Length(array);

array2 = new string[i];

Array.Copy(array, array2, i);

do

{

Shuffle(random, array2);

} while (!CompararCircular(array, array2));

ToString(array, array2);

return array;

}

//esta funcion lee el numero que el usuario mete para saber si esta en el rango (1-3) o no es un numero

static int RespuestaNumero()

{

int respuesta;

bool verificacion = int.TryParse(Console.ReadLine(), out respuesta);

do

{

if(respuesta > 3 || respuesta < 1|| !verificacion)

{

Console.WriteLine("\t\t\t\tOpción inválida, número entero entre el 1 y el 3");

verificacion = int.TryParse(Console.ReadLine(), out respuesta);

}

} while (respuesta > 3 || respuesta < 1 || !verificacion);

return respuesta;

}

//esta funcion lee el numero que el usuario mete para saber si esta en el rango (1-4) o no es un numero

static int RespuestaNumeroParejas()

{

int respuesta;

bool verificacion = int.TryParse(Console.ReadLine(), out respuesta);

do

{

if (respuesta == 0 || respuesta > 4 || respuesta < 1 || !verificacion)

{

Console.WriteLine("\t\t\t\tOpción inválida, número entero entre el 1 y el 4");

verificacion = int.TryParse(Console.ReadLine(), out respuesta);

}

} while (respuesta == 0 || respuesta > 4 || respuesta < 1 || !verificacion);

return respuesta;

}

//esta funcion lee el numero que el usuario mete para saber si esta en el rango (1-5) o no es un numero

static int RespuestaNumeroCicular()

{

int respuesta;

bool verificacion = int.TryParse(Console.ReadLine(), out respuesta);

do

{

if (respuesta == 0 || respuesta > 5 || respuesta < 1 || !verificacion)

{

Console.WriteLine("\t\t\t\tOpción inválida, número entero entre el 1 y el 5");

verificacion = int.TryParse(Console.ReadLine(), out respuesta);

}

} while (respuesta == 0 || respuesta > 5 || respuesta < 1 || !verificacion);

return respuesta;

}

//esta funcion logra organizar las funciones necesarias para que el modo basico sirva

static void ModoBasico()

{

Console.Clear();

//llamamos al menu basico

MenuBasico();

//utilizamos el numero que el usuario introduce

int q = RespuestaNumero();

//inicializamos los arrays que necesitamos

string[] personas = new string[30];

string[] personas2 = new string[2];

int i;

Random random = new Random();

//para saber si ya fue randomziado o no

bool randomized = false;

while (q < 3)

{

switch (q)

{

//caso de introducir numero

case 1:

i = Length(personas);

//verificamos si ya fue randomizado o no

if (i <= 30 && randomized)

{

string[] temp = IntroducirNombreRandomizado(personas, personas2);

Array.Copy(temp, personas, temp.Length - 1);

}

else if (i <= 30)

{

string[] temp = IntroducirNombre(personas);

Array.Copy(temp, personas, temp.Length - 1);

}

//si llegamos al limite nos avisa

else

{

Console.WriteLine("\t\t\t\tMáxima cantidad de personas admitidas");

}

break;

//caso de randomizar

case 2:

i = Length(personas);

//verificamos que hayan mas de dos personas para poder de verdad randomizar

if (i >= 2)

{

personas2 = new string[i];

Array.Copy(personas, personas2, i);

//se hace el random hasta que comparar nos de verdadero

do

{

Shuffle(random, personas2);

} while (!Comparar(personas, personas2));

ToString(personas, personas2);

//decimos que ya fue randomziado

randomized = true;

}

else

{

Console.WriteLine("\t\t\t\tNo se puede randomizar si no hay al menos 2 personas");

}

break;

// sale del bucle

case 3:

break;

}

Console.ReadKey();

Console.Clear();

MenuBasico();

q = RespuestaNumero();

}

}

//esta funcion logra organizar las funciones necesarias para que el modo parejas sirva

static void ModoParejas()

{

Console.Clear();

MenuPareja();

//llamamos el menu de este modo y la funcion que admite este rango

int q = RespuestaNumeroParejas();

string[] personas = new string[30];

string[] personas2 = new string[2];

int i= 0;

Random random = new Random();

bool randomized = false;

while (q < 4)

{

switch (q)

{

case 1:

i = Length(personas);

//verficamos si ya ha sido randomizado o no

if (i <= 30 && randomized)

{

string[] temp = IntroducirNombreRandomizadoPareja(personas, personas2);

Array.Copy(temp, personas, temp.Length - 1);

}

else if (i <= 30)

{

string[] temp = IntroducirNombre(personas);

Array.Copy(temp, personas, temp.Length - 1);

}

else

{

Console.WriteLine("\t\t\t\tMáxima cantidad de personas admitidas");

}

break;

case 2:

i = Length(personas);

//ya que introducimos parejas, el valor limite no puede ser 29 sino 28 porque se agregan de 2 en 2 y tambien verifica si ya fue randomizado o no

if (i < 28 && randomized)

{

string[] vs = IntroducirParejaRandomizado(personas, personas2);

Array.Copy(vs, personas,vs.Length - 1);

}

else if(i <= 28)

{

string[] temporal = IntroducirPareja(personas);

Array.Copy(temporal, personas, temporal.Length - 1);

}

else

{

Console.WriteLine("\t\t\t\tMáxima cantidad de personas admitidas");

}

break;

case 3:

i = Length(personas);

//vemos si el array tiene mas de 2 personas para ver si el shuffle se podra llevar a cabo

if (i >= 2)

{

personas2 = new string[i];

Array.Copy(personas, personas2, i);

//hacemos el shuffle hasta que el comparar nos diga que es verdadero

do

{

Shuffle(random, personas2);

} while (!CompararParejas(personas, personas2));

ToString(personas, personas2);

//hacemos que el randomizado ha sido verdadero

randomized = true;

}

else

{

Console.WriteLine("\t\t\t\tNo se puede randomizar si no hay al menos 2 personas");

}

break;

//sale del menu

case 4:

break;

}

Console.ReadKey();

Console.Clear();

MenuPareja();

q = RespuestaNumeroParejas();

}

}

//esta funcion logra organizar las funciones necesarias ara que el modo circular sirva

static void ModoCircular()

{

Console.Clear();

MenuCircular();

//utilizamos el modo circular que nos conviene por el rango

int q = RespuestaNumeroCicular();

string[] personas = new string[30];

string[] personas2 = new string[2];

int i = 0;

Random random = new Random();

bool randomized = false;

bool circular = false;

while (q < 5)

{

switch (q)

{

case 1:

i = Length(personas);

// introducimos numeor y comparamos primero si ha sido randomizado por circular o por parejas

if (i <= 30 && circular)

{

string[] temp = IntroducirNombreRandomizadoCircular(personas, personas2);

Array.Copy(temp, personas, temp.Length - 1);

}

else if (i <= 30 && randomized)

{

string[] temp = IntroducirNombreRandomizadoPareja(personas, personas2);

Array.Copy(temp, personas, temp.Length - 1);

}

else if (i <= 30)

{

string[] temp = IntroducirNombre(personas);

Array.Copy(temp, personas, temp.Length - 1);

}

else

{

Console.WriteLine("\t\t\t\tMáxima cantidad de personas admitidas");

}

break;

case 2:

//para introducir parejas es lo mismo, verificamos si ha sido randomizaod por circular o por parejas

i = Length(personas);

if (i < 28 && circular)

{

string[] vs = IntroducirParejaRandomizadoCircular(personas, personas2);

Array.Copy(vs, personas, vs.Length - 1);

}

else if (i < 28 && randomized)

{

string[] vs = IntroducirParejaRandomizado(personas, personas2);

Array.Copy(vs, personas, vs.Length - 1);

}

else if (i <= 28)

{

string[] temporal = IntroducirPareja(personas);

Array.Copy(temporal, personas, temporal.Length - 1);

}

else

{

Console.WriteLine("\t\t\t\tMáxima cantidad de personas admitidas");

}

break;

case 3:

//se randomiza por parejas

i = Length(personas);

if (i >= 2)

{

personas2 = new string[i];

Array.Copy(personas, personas2, i);

do

{

Shuffle(random, personas2);

} while (!CompararParejas(personas, personas2));

ToString(personas, personas2);

randomized = true;

}

else

{

Console.WriteLine("\t\t\t\tNo se puede randomizar si no hay al menos 2 personas");

}

break;

case 4:

//se randomiza por circular

i = Length(personas);

if (i >= 2)

{

personas2 = new string[i];

Array.Copy(personas, personas2, i);

//se hace el shuffle circulaar hasta que comparar nos diga que se logro

do

{

string[] temp = ShuffleCircula(random, personas2);

Array.Copy(temp, personas2, temp.Length);

//ya que nos devuelve un string nuestra funcion, lo guardamos en un temporal y despues la copiamos en el array deseado

} while (!CompararCircular(personas, personas2));

ToString(personas, personas2);

circular = true;

}

else

{

Console.WriteLine("\t\t\t\tNo se puede randomizar si no hay al menos 2 personas");

}

break;

case 5:

break;

}

Console.ReadKey();

Console.Clear();

MenuCircular();

q = RespuestaNumeroCicular();

}

}

//esta funcion imprime el menu principal y hace que utilicemos el modo que querramos

static void ModoEmpezar()

{

MenuPrincipal();

int q = RespuestaNumero();

switch (q)

{

case 1:

ModoBasico();

break;

case 2:

ModoParejas();

break;

case 3:

ModoCircular();

break;

}

}

static void Main(string[] args)

{

ModoEmpezar();

}

}

}