Introducción a la Computación (para Matemáticas)

Primer Cuatrimestre de 2018

DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Escorpión

El juego del Escorpión consiste en lo siguiente: un jugador genera un número de cuatro dígitos (sin repetir) que su oponente debe adivinar. Para ello el segundo jugador propone un número y el primero responde indicando cuantos dígitos enunciados están en la posición correcta (buenos) y cuantos dígitos de los enunciados coinciden con uno de los originales pero en una ubicación incorrecta (regulares). Por ejemplo si el número generado es [4 5 6 7] y el oponente propone [4 5 7 8], la respuesta del primero será “2 bien, 1 regular”. El adivinador tiene diez oportunidades para lograr su objetivo. Sino lo logra falla.

Ejercicio

Escribir un programa que implemente el juego del escorpión. Para ello el programa debe:

a. Generar la secuencia secreta de cuatro dígitos distintos.
b. Solicitar al usuario cuatro dígitos distintos.
c. Informar al usuario sus aciertos.
d. Sino acertó y lo intentó menos de diez veces volver al paso b.

Ejercicio (cont.)

Tareas:

1. Generar número de cuatro dígitos distintos.
2. Pedir número de cuatro dígitos distintos.
3. Comparar dos números de igual cantidad de dígitos.
4. Jugar.

Cuatro dígitos distintos

Especificación:
Función booleana que dado un vector/arreglo de cuatro dígitos me devuelve true si no hay valores repetidos y false en caso contrario.

Por ejemplo:

a. sin_digitos_repetidos({1, 2, 3, 4}) es true.
b. sin_digitos_repetidos({2,3,4,2}) es false.

Algoritmo:

Cuatro dígitos distintos

```
bool sin_digitos_repetidos(array<int,4> a){
    bool SI = true;
    int i=0;
    int j=1;
    while(SI && i<4){
        while(j<4){
            if (a[i]==a[j]) {SI=false;}
            j=j+1;
        }
        i=i+1;
        j=i+1;
    }
    return SI;
}
```

Generar

```
// Genera un arreglo de 4 dígitos al azar entre 0 y 9
array<int,4> generar()
{array<int,4> azar ={0,0,0,0};
    int i=0;
    int seed = time(NULL);
    srand(seed);
    while(i<4) {
        azar[i] = rand() % 10;
        i=i+1;
    }
    return azar;
}
```

Testear generar

```
int main() {
int i =0;
array<int,4> a =generar();
while(i<4) {
    cout << a[i]  << endl;
    i=i+1;
}
```

Generar sin repetir

Ya sabemos generar un vector de cuatro dígitos al azar y sabemos como chequear que dicho vector no tiene repetidos.
¿Cómo los combinamos?

```
array<int,4> generar_sin_repetir(){
    array<int,4> Sin_repe ={0,0,0,0};
    bool Sin=false;
    while (!Sin){
        Sin_repe=generar();
        Sin=sin_digitos_repetidos(Sin_repe);
    }
     return Sin_repe;
}
```

Generar sin repetir (cont.)

Notar que la versión que acabamos de escribir permite un cero en la primera componente.
¿Cómo evitarlo?

```
array<int,4> generar_sin_repetir(){
    array<int,4> Sin_repe ={0,0,0,0};
    bool Sin=false;
    while (!(Sin && Sin_repe[0]!=0)){
        Sin_repe=generar();
        Sin=sin_digitos_repetidos(Sin_repe);
    }
     return Sin_repe;
}
```

Testear generar sin repetir

```
int main() {
int i =0;
array<int,4> a =generar_sin_repetir();
while(i<4) {
    cout << a[i]  << endl;
    i=i+1;
}
```

Pedir

```
array<int,4> pedir(){
    array<int,4> propuesto={0,0,0,0};
    int i =0;
    bool Sin_repe=false;
    while(!Sin_repe){
        while(i<4){
            cout << "Ingrese el " << i << " -esimo dígito" << endl;
            cin >> propuesto[i];
        i=i+1;
        }
        if(sin_digitos_repetidos(propuesto)){Sin_repe=true;}
        else {
            cout << “Dígitos repetidos. Tendrá que ingresarlos de nuevo" << endl;
        }
        i=0;
    }
    return propuesto;
}
```

Testear pedir

```
int main(){
    array<int,4> a =pedir();
    int i=0;
    while(i<4) {
        cout << a[i] << endl;
        i = i + 1;
    }
}
```

Comparar

Especificación:
Dado dos vectores de 4 cuatro posiciones con un dígito distinto en cada una de ellas, determinar cuantos buenos y cuantos regulares tienen.

¿ES CORRECTA?

Comparar

Especificación 2:
Dado dos vectores de 4 cuatro posiciones con un dígito distinto en cada una de ellas, asumiendo que el primero tiene un patrón que debe ser comparado con el segundo para determinar cuantos buenos y cuantos regulares tiene este respecto del primero.

¿AHORA ES CORRECTA?

Comparar

Especificación 3:
Dado dos vectores de 4 cuatro posiciones con un dígito distinto en cada una de ellas, asumiendo que el primero tiene un patrón que debe ser comparado con el segundo para determinar cuantos de sus dígitos son buenos y cuantos de ellos son regulares, respecto del primero.
Entendiendo por dígito bueno aquel valor que encontrándose en la posición i-esima del segundo vector coincide en valor con el dígito de la possición i-esima del primer vector.
El valor de un dígito del segundo vector es considerado regular si dicho valor también aparece en el primer vector pero en una posición distinta.

Comparar

Algoritmo:
Ejemplos:

a.{1,2,3,4} y {1.2,3,4}
b.{1,2,3,4} y {4,2,3,1}
c.{1,2,3,4} y {4,2,6,1}
d.{1,2,3,4} y {7,8,6,1}

Comparar

```
array<int,2> comparar(array<int,4> secreto,array<int,4> propuesto){
    int i=0;
    int j=0;
    array<int,2> aux = {0,0};
    while(i<4){
        if (propuesto[i]==secreto[i]) {aux[0]=aux[0]+1;}
        else{
            while(j<4){
                if (i != j) {
                    if (propuesto[i]==secreto[j]) {aux[1]=aux[1]+1;
                    }
                }
                j=j+1;
            }
        }
        i=i+1;
        j=0;
    }
    return aux;
}
```

Testear comparar

```
int main(){
array<int,4> s={1,2,3,4};
array<int,4> p={1,3,2,4};
array<int,2> a = {0,0};
a= comparar(s,p);
cout << "Tiene buenos " << a[0] << endl;
cout << "Tiene regulares " << a[1] << endl;
return 0;
}
```

Jugar

Ahora estamos en condiciones de combinar todas las funciones para resolver el problema original.

Jugar

```
void jugar(){
    int jugada=0;
    bool gano=false;
    array<int,4> secreto=generar_sin_repetir();
    array<int,4> propuesto={0,0,0,0};
    array<int,2> Buenos_Regulares = {0,0};
    while(jugada < 10 || gano){
        propuesto=pedir();
        Buenos_Regulares=comparar(secreto,propuesto);
        if(Buenos_Regulares[0]==4){gano=true;}
        jugada=jugada+1;
    }
    if(gano){
        cout << "GANOOOOOOO!!!!" << endl;
    } else{
        cout << "PERDIO :-(" << endl;
    };
}
```

```
    bool gano=false;
    array<int,4> secreto=generar_sin_repetir();
    array<int,4> propuesto={0,0,0,0};
    array<int,2> Buenos_Regulares = {0,0};
    while(jugada < 10 && !gano){
        propuesto=pedir();
        Buenos_Regulares=comparar(secreto,propuesto);
        if(Buenos_Regulares[0]==4){gano=true;}
        else{
            cout << "Su número propuesto: " << propuesto[0] <
            cout << "Tiene buenos " << Buenos_Regulares[0] <<
            cout << "Tiene regulares " << Buenos_Regulares[1]
            cout << "Intentelo nuevamente." << endl;
        };
        jugada=jugada+1;
    }
    if(gano){
        cout << "GANOOOOOOO!!!!" << endl;
    } else{
        cout << "PERDIO :-(" << endl;
    };

}
```

Repaso de la clase de hoy

- Modularidad del código: funciones y procedimientos.
- Composición de funciones.

Próximos temas

- Especificación de problemas.
- Correctitud de programas.