UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

Ingeniería en Software y Tecnologías Emergentes

Programación Estructurada

Actividad 8. Parte 1

ALUMNO: Fernando Haro Calvo

MATRICULA: 372106

GRUPO: 932

PROFESOR: Pedro Núñez Yépiz

2 de octubre del 2023

Funciones Auxiliares:

```
97  //**********************
98  // Valida la entrada del usuario en un rango de numeros.
99  int validar(char msg[], int ri, int rf)
100 {
101     char cadena[50];
102     int num;
103 
104     do
105     {
106         printf("%s", msg);
107         fflush(stdin);
108         gets(cadena);
109         num = atoi(cadena);
110     } while (num < ri || num > rf);
111 
112     return num;
113 }
```

Ejercicio 1:

```
115  //**********************
116  // Llena el vector con valores introducidos por el usuario del 30 al 70.
117  // HCF_ACT8_01_932
118  void llenarVectManual(int vector[], int m)
119  {
120      int i, num;
121
122      for (i = 0; i < m; i++)
123      {
124          printf("Valor %d -> ", i + 1);
125          num = validar("Ingrese un numero entre 30-70: ", 30, 70);
126          vector[i] = num;
127      }
128  }
```

Salida:

```
Valor 1 -> Ingrese un numero entre 30-70: 30
Valor 2 -> Ingrese un numero entre 30-70: 35
Valor 3 -> Ingrese un numero entre 30-70: 40
Valor 4 -> Ingrese un numero entre 30-70: 45
Valor 5 -> Ingrese un numero entre 30-70: 50
Valor 6 -> Ingrese un numero entre 30-70: 55
Valor 7 -> Ingrese un numero entre 30-70: 60
Valor 8 -> Ingrese un numero entre 30-70: 65
Valor 9 -> Ingrese un numero entre 30-70: 70
Valor 10 -> Ingrese un numero entre 30-70: 30


Presione una tecla para continuar . . .
```

Ejercicio 2:

```
130  //*********************
131  // Llena el vector con valores generados aleatoriamente del 1 al 20 sin repetir.
132  // HCF_ACT8_02_932
133  void llenarVectAuto(int vector[], int m)
134  {
135      int i, j, num, band;
136
137      for (i = 0; i < m; i++)
138      {
139          do
140          {
141              band = 0; // Reinicia bandera
142              num = rand() % 20 + 1;
143
144              // Validar en cada espacio el numero
145              for (j = 0; j < i; j++)
146              {
147                  if (vector[j] == num)
148                  {
149                      band = 1; // Activa bandera
150                  }
151              }
152          } while (band != 0);
153
154          vector[i] = num;
155      }
156
157      printf("Vector llenado con exito!");
158  }
```

Salida:

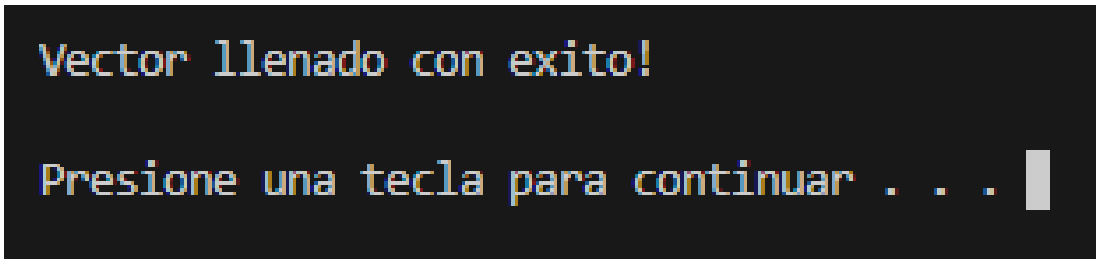
```
Vector llenado con exito!

Presione una tecla para continuar . . .
```

Ejercicio 3:

```
160  //**********************
161  // Llena un vector con los valores de otros dos vectores.
162  // HCF_ACT8_03_932
163  void llenarVectConVectores(int vector1[], int m, int vector2[], int n, int vector3[], int p)
164  {
165      int i;
166
167      // Validar si existen suficientes numeros para llenar el vector.
168      if (m < p / 2 || n < p / 2)
169      {
170          printf("Los vectores no son suficientemente grandes para llenar el vector");
171          system("PAUSE");
172          return;
173      }
174
175      // Llenar vector3 con vector 1 y 2.
176      for (i = 0; i < p / 2; i++)
177      {
178          vector3[i] = vector1[i];
179          vector3[p / 2 + i] = vector2[i];
180      }
181
182      printf("Vector llenado con exito!");
183  }
```

Salida:



Ejercicio 4:

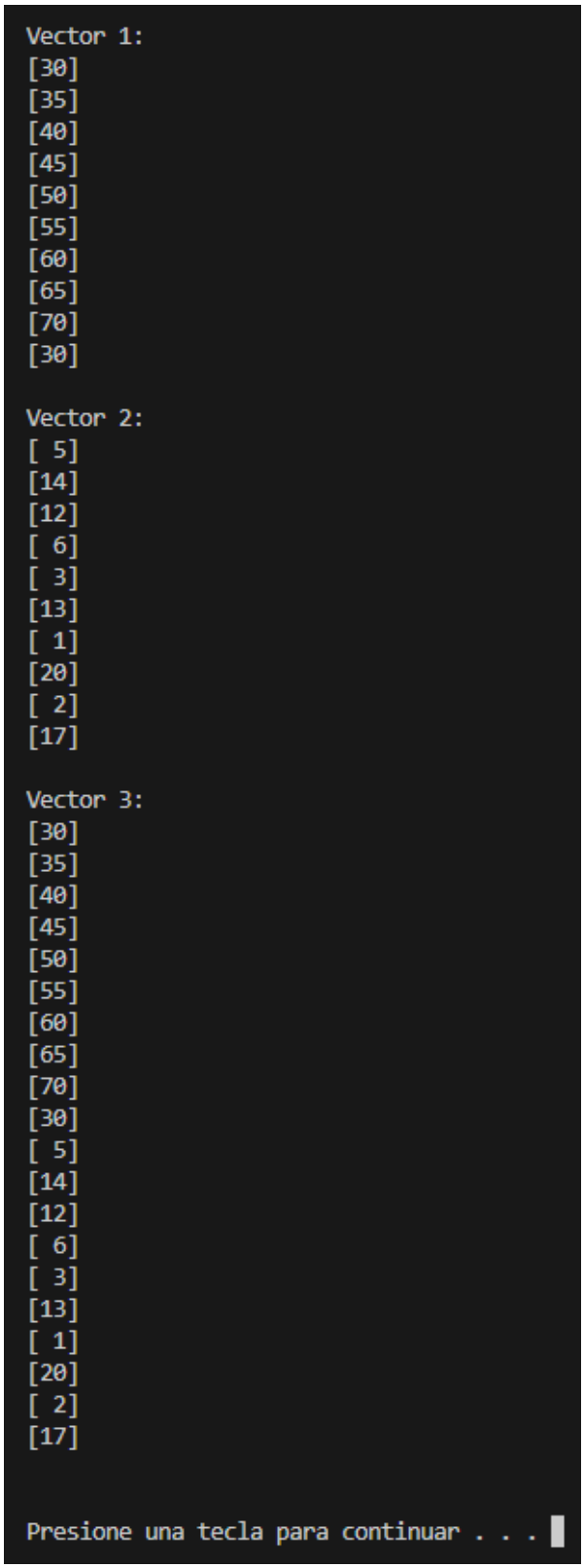
```c
//**********************
// Imprime los valores de todos los vectores.
// HCF_ACT8_04_932
void imprimirVectores(int vector1[], int m, int vector2[], int n, int vector3[], int p)
{
    int i;

    printf("Vector 1:\n");
    for (i = 0; i < m; i++)
    {
        printf("[%2d]\n", vector1[i]);
    }

    printf("\nVector 2:\n");
    for (i = 0; i < n; i++)
    {
        printf("[%2d]\n", vector2[i]);
    }

    printf("\nVector 3:\n");
    for (i = 0; i < p; i++)
    {
        printf("[%2d]\n", vector3[i]);
    }
}
```

Salida:


Vector 1:
[30]
[35]
[40]
[45]
[50]
[55]
[60]
[65]
[70]
[30]
Vector 2:
[5]
[14]
[12]
[6]
[3]
[13]
[1]
[20]
[2]
[17]
Vector 3:
[30]
[35]
[40]
[45]
[50]
[55]
[60]
[65]
[70]
[30]
[5]
[14]
[12]
[6]
[3]
[13]
[1]
[20]
[2]
[17]
Presione una tecla para continuar . . .

Ejercicio 5:

```
211  //**********************
212  // Llena una matriz 4x4 con los datos de dos vectores.
213  // HCF_ACT8_05_932
214  void llenarMatriz4x4(int vector1[], int m, int vector2[], int n, int matriz[][4])
215  {
216      int i, j, cont;
217
218      for (i = 0, cont = 0; i < 4; i++)
219      {
220          for (j = 0; j < 4; j++, cont++)
221          {
222              if (cont < m) // Mientras vector1 tenga datos
223              {
224                  matriz[i][j] = vector1[cont];
225              }
226              else if (cont < (m + n)) // Mientras vector2 tenga datos
227              {
228                  matriz[i][j] = vector2[cont - m];
229              }
230              else // Ambos vectores vacios, asignar 0.
231              {
232                  matriz[i][j] = 0;
233              }
234          }
235      }
236
237      printf("Matriz llenada con exito!");
238  }
```

Salida:

```
Matriz llenada con exito!

Presione una tecla para continuar . . .
```

Ejercicio 6:

```
240   //**********************
241   // Imprime los valores de la matriz 4x4.
242   // HCF_ACT8_06_932
243   void imprimirMatriz(int matriz[][4])
244   {
245       int i, j;
246
247       for (i = 0; i < 4; i++)
248       {
249           for (j = 0; j < 4; j++)
250           {
251               printf("[%2d]", matriz[i][j]);
252           }
253           printf("\n");
254       }
255   }
```

Salida:

```
[30][35][40][45]
[50][55][60][65]
[70][30][ 5][14]
[12][ 6][ 3][13]


Presione una tecla para continuar . . .
```