

Fundamentos de Programación

PEC4 - 20181

Fecha límite de entrega: 22/10/2018

Estudiante

Apellidos: de Lima B. Campos

Nombre: Lidiane A.

Objetivos

- Saber aplicar correctamente la estructura de control iterativa.
- Aprender a utilizar el tipo de datos vector para representar estructuras de datos sencillos.
- Saber utilizar las funciones de conversión de tipo cuando haga falta.

Formato y fecha de entrega

La PEC se debe entregar antes del día **22 de octubre de 2018 a las 23:59**.

Para la entrega se deberá entregar un fichero en formato **ZIP**, que contenga:

- Este documento con la respuesta del ejercicio 1 y capturas de pantalla de las pruebas del ejercicio 2.
- Un proyecto Codelite que contenga los ficheros .c pedidos al ejercicio 2

Hay que hacer la entrega en el apartado de entregas de AC del aula de teoría.

Enunciado

Siguiendo con la ayuda que proporcionamos a la compañía UOCRailways, nos han pedido nuestra colaboración para crear un programa que los ayude a gestionar los trenes. En los siguientes ejercicios, trabajaremos con vectores, las estructuras iterativas y la entrada y salida interactiva para gestionar los datos de los trenes.

Dados los siguientes tipos de datos y variables definidos para gestionar los datos de varios trenes:

Lenguaje algorítmico

const

MAX_TRAINS: integer = 10;

MAX_TRAIN_LENGTH: integer = 450;

end const

type

tPropulsion = { ELECTRICAL, CARBON, SOLAR, GASOLINE, DIESEL,
MAGNETIC };

end type

var

idTrain: **vector**[MAX_TRAINS] of **integer**;

locLength: **vector**[MAX_TRAINS] of **integer**;

```
propulsionType: vector[MAX_TRAINS] of tPropulsion;  
wagonLength: vector[MAX_TRAINS] of real;  
spaceBetweenVagons: vector[MAX_TRAINS] of real;  
nWagons: vector[MAX_TRAINS] of integer;  
end var
```

Ejercicio 1: [50%]

Implementar en lenguaje algorítmico, un algoritmo que permita introducir de manera interactiva por el canal estándar de entrada los datos de varios trenes (hasta el máximo permitido) en los vectores que tenéis declarados en el enunciado. Al finalizar, el algoritmo tiene que mostrar por el canal estándar de salida el contenido de los vectores.

El funcionamiento del algoritmo es el siguiente:

- Se pide al usuario que introduzca el número de trenes que se quieren introducir en los correspondientes vectores.
- En caso de que el número de trenes a introducir sea válido (superior a cero e inferior o igual al número máximo permitido), se continúa adelante.
- En caso de que el número de trenes NO sea válido, se muestra un mensaje de error por el canal estándar de salida y se vuelve a solicitar el número de trenes que se quieren introducir.
- El proceso anterior se repite indefinidamente hasta que el usuario introduzca un número de trenes válido.
- Una vez introducido un número de trenes válido, el algoritmo pide por el canal estándar de entrada que se introduzcan los datos de cada uno de ellos (*idTrain*, *locLength*, *propulsionType*, *wagonLength*, *spaceBetweenVagons* y *nWagons*), y los guarda dentro del vector correspondiente.
- Los datos introducidos de los trenes se pueden considerar correctos. Por ejemplo, cuando se introduzca *idTrain* (el identificador de un tren), se da por supuesto que el usuario introducirá un valor entero, etc.
- Una vez introducidos los datos de un tren, el algoritmo calcula su longitud, con los datos disponibles (longitud de la locomotora, longitud de los vagones, y espacio entre vagones), y la guarda dentro de la variable del vector correspondiente.
- Con la longitud calculada, el algoritmo comprueba si el tren supera la longitud máxima (definida por una constante), y guarda un valor booleano (cierto o falso) dentro de la variable del vector correspondiente. NO HAY que efectuar ninguna otra comprobación ni validación, sólo guardar si la longitud del tren supera la máxima.

- Para acabar, el algoritmo tiene que mostrar por el canal estándar de salida el contenido de los vectores, con texto que explique qué es cada valor mostrado.

Observaciones:

- Hay que utilizar estructuras iterativas, en sus diversas variantes, según la que sea la más útil en cada caso.
- Hay que declarar los vectores auxiliares necesarios y las variables necesarias donde guardar las longitudes de los trenes y si dichas longitudes superan o no la longitud máxima.
- Recordad que, como en las PEC anteriores, en el caso del lenguaje algorítmico podéis utilizar la función `readPropulsionType()` y `writePropulsionType()` para leer y escribir el tipo de propulsión y `readBoolean()` y `writeBoolean()` para leer y escribir booleanos.

Ejercicio 2: [50%]

Implementar en lenguaje C, el algoritmo del ejercicio anterior. Como en las anteriores PECs es pide que mostréis el funcionamiento del algoritmo **haciendo tres capturas de pantalla** donde se vea la salida de vuestro programa de los valores de trenes que se han introducido en los vectores. Las capturas de pantallas las tenéis que adjuntar en el documento donde hay la respuesta del ejercicio 1.

Criterios de corrección:

- Que se siga la notación algorítmica utilizada a la asignatura. Ved documento *Nomenclator* a la xWiki de contenido.
- Que se respeten los criterios de estilo de programación C. Ved la *Guía de estilo de programación en C* que tenéis a la Wiki de contenido.
- Que el programa se adecúe a las indicaciones dadas.
- Que el programa compile y funcione de acuerdo con el que se pide.
- Que se declaren los tipos adecuados según el tipo de datos que representa.

Algorithm *bucles*

const

```
MAX_TRAINS: integer = 10;
MAX_TRAIN_LENGTH: integer = 450;
```

end const

type

tPropulsion = { ELECTRICAL, CARBON, SOLAR, GASOLINE, DIESEL,
MAGNETIC };

end type

var

isOverMaxTrainLength: **boolean**;
overMaxTrainLength: **vector**[trainQuantity] of **integer**;
i: **integer**;
trainQuantity: **integer**;
idTrain: **vector**[trainQuantity] of **integer**;
locLength: **vector**[MAX_TRAINS] of **integer**;
propulsionType: **vector**[MAX_TRAINS] of **tPropulsion**;
wagonLength: **vector**[MAX_TRAINS] of **real**;
spaceBetweenVagons: **vector**[MAX_TRAINS] of **real**;
nWagons: **vector**[MAX_TRAINS] of **integer**;
trainLength: **vector**[trainQuantity] of **integer**;

end var

writeString("Enter quantity of trains: ");

trainQuantity := readInteger();

while trainQuantity \leq 0 or trainQuantity > MAX_TRAINS **do**

writeString("Enter correct quantity of trains it must be between 1 and
10: ");

trainQuantity := readInteger();

end while

```

for i:=1 to trainQuantity do
  writeString("Enter the train id: ");
  idTrain[i]:=readInteger();

  writeString("Enter the locomotive's length ");
  locLength[i]:= readInteger();

  writeString("Enter the propulsion type ("1-ELECTRICAL, 2-CARBON, 3-SOLAR,
  4-GASOLINE, 5-DIESEL, 6-MAGNETIC ) ");
  propulsionType[i]:= readPropulsionType();

  writeString("Enter the wagon's Length: ");
  wagonLength[i]:= readReal();

  writeString("Enter the space between the wagons: ");
  spaceBetweenVagons[i]:= readReal();

  writeString("Enter the number of Wagons: ");
  nWagons[i]:= readInteger();

  trainLength[i]:= (locLength[i] + wagonLength[i] +(spaceBetweenVagons[i] *
  nWagons[i]) +nWagons[i]);

end for

```

```

for j:=1 to trainQuantity do
    if trainLength[j] > MAX_TRAIN_LENGTH then
        overMaxTrainLength[j]:= 1;
    else
        overMaxTrainLength[j]:= 0;
    end if

end for

for k=1 to trainQuantity do
    writeString("Train id: ");
    writeInteger("%d\n", idTrain[k]);

    writeString("The locomotive's length ");
    writeInteger("%d\n", locLength[k]);

    writeString("The propulsion type (1-ELECTRICAL, 2-CARBON, 3-SOLAR, 4-
    GASOLINE, 5-DIESEL, 6-MAGNETIC ) ");
    writePropulsionType("%u\n", propulsionType[k]);

    writeString("The wagon's Length: ");
    writeReal("%f\n", wagonLength[k]);

    writeString("The space between the wagons: ");

```

```
writeReal( "%f\n", spaceBetweenVagons[k]);
```

```
writeString("The number of Wagons: ");
```

```
writeString("%d\n", nWagons[k]);
```

```
writeString("The Train length: ");
```

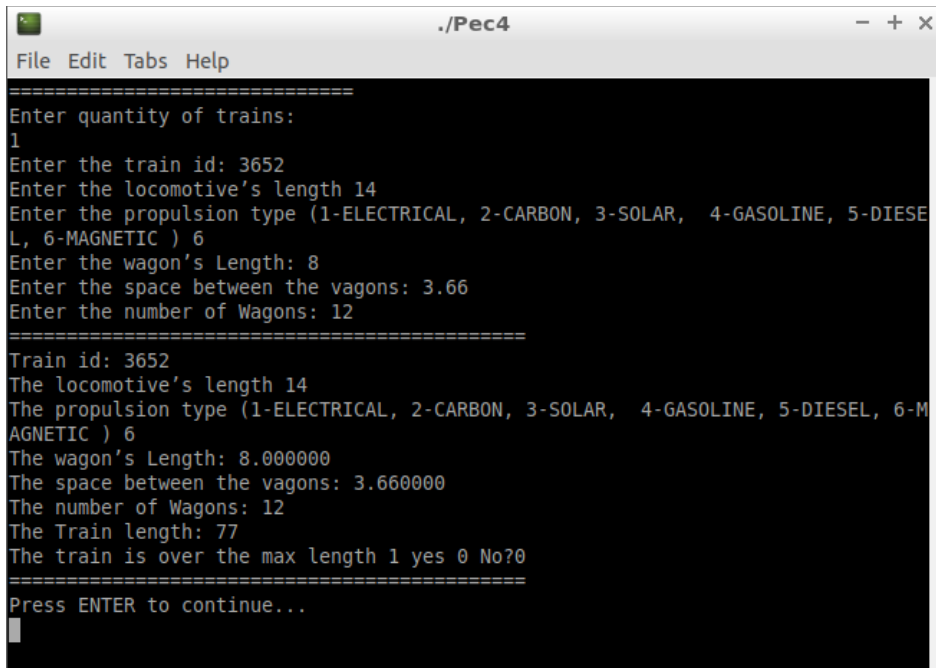
```
writeInteger("%d\n", trainLength[k]);
```

```
writeString("The train is over the max length 1 yes 0 No?");
```

```
writeBoolean("%u\n", overMaxTrainLength[k] );
```

end for

end algorithm



```
./Pec4
File Edit Tabs Help
=====
Enter quantity of trains:
1
Enter the train id: 3652
Enter the locomotive's length 14
Enter the propulsion type (1-ELECTRICAL, 2-CARBON, 3-SOLAR, 4-GASOLINE, 5-DIESEL, 6-MAGNETIC ) 6
Enter the wagon's Length: 8
Enter the space between the wagons: 3.66
Enter the number of Wagons: 12
=====
Train id: 3652
The locomotive's length 14
The propulsion type (1-ELECTRICAL, 2-CARBON, 3-SOLAR, 4-GASOLINE, 5-DIESEL, 6-MAGNETIC ) 6
The wagon's Length: 8.000000
The space between the wagons: 3.660000
The number of Wagons: 12
The Train length: 77
The train is over the max length 1 yes 0 No?0
=====
Press ENTER to continue...
█
```



```
./Pec4
File Edit Tabs Help
Enter the locomotive's length 20
Enter the propulsion type (1-ELECTRICAL, 2-CARBON, 3-SOLAR, 4-GASOLINE, 5-DIESEL, 6-MAGNETIC ) 2
Enter the wagon's Length: 5
Enter the space between the wagons: 2
Enter the number of Wagons: 10
Enter the train id: 2
Enter the locomotive's length 10
Enter the propulsion type (1-ELECTRICAL, 2-CARBON, 3-SOLAR, 4-GASOLINE, 5-DIESEL, 6-MAGNETIC ) 3
Enter the wagon's Length: 10
Enter the space between the wagons: 5
Enter the number of Wagons: 23
Train id: 1
The locomotive's length 20
The propulsion type (1-ELECTRICAL, 2-CARBON, 3-SOLAR, 4-GASOLINE, 5-DIESEL, 6-MAGNETIC ) 2
The wagon's Length: 5.000000
The space between the wagons: 2.000000
The number of Wagons: 10
The Train length: 55
The train is over the max length 1 yes 0 No?0
Train id: 2
The locomotive's length 10
The propulsion type (1-ELECTRICAL, 2-CARBON, 3-SOLAR, 4-GASOLINE, 5-DIESEL, 6-MAGNETIC ) 3
The wagon's Length: 10.000000
The space between the wagons: 5.000000
The number of Wagons: 23
The Train length: 158
The train is over the max length 1 yes 0 No?0
Press ENTER to continue...
```

```
./Pec4
File Edit Tabs Help
=====
Enter quantity of trains:
2
Enter the train id: 12
Enter the locomotive's length 10
Enter the propulsion type (1-ELECTRICAL, 2-CARBON, 3-SOLAR, 4-GASOLINE, 5-DIESEL, 6-MAGNETIC ) 4
Enter the wagon's Length: 10
Enter the space between the wagons: 2.5
Enter the number of Wagons: 6
Enter the train id: 25
Enter the locomotive's length 12
Enter the propulsion type (1-ELECTRICAL, 2-CARBON, 3-SOLAR, 4-GASOLINE, 5-DIESEL, 6-MAGNETIC ) 5
Enter the wagon's Length: 12
Enter the space between the wagons: 2
Enter the number of Wagons: 10
Train id: 12
The locomotive's length 10
The propulsion type (1-ELECTRICAL, 2-CARBON, 3-SOLAR, 4-GASOLINE, 5-DIESEL, 6-MAGNETIC ) 4
The wagon's Length: 10.000000
The space between the wagons: 2.500000
The number of Wagons: 6
The Train length: 41
The train is over the max length 1 yes 0 No?0
=====
Train id: 25
The locomotive's length 12
The propulsion type (1-ELECTRICAL, 2-CARBON, 3-SOLAR, 4-GASOLINE, 5-DIESEL, 6-MAGNETIC ) 5
The wagon's Length: 12.000000
The space between the wagons: 2.000000
The number of Wagons: 10
The Train length: 54
The train is over the max length 1 yes 0 No?0
=====
Press ENTER to continue...
```