Actividad Integral estructura de datos lineales (Evidencia Competencia)

Tecnológico de Monterrey

Miguel Jiménez Padilla - A01423189

Profesora:

Mónica Larre Bolaños

09 de Octubre de 2021

Casos de prueba:

El primer caso de prueba es una validación para el nombre del archivo, si el archivo no tiene nada o no existe tienes que volverlo a escribir

```
> clang++-7 -pthread -std=c++17 -o main main.cpp
> ./main
Enter the file name: abcd

The list is empty
The list is empty
The file is empty or doesn't exist, try again
Enter the file name:
```

```
Enter the file name: canal.txt

Main:
A: Search for UBI Coincidences
B: Exit
```

Posteriormente, se despliega un menú para buscar coincidencias de UBI, o para salir.

```
Main:
A: Search for UBI Coincidences
B: Exit
A

-------------------------------------------
Enter the first 3 caracters of the UBI: 0AN
-------------------------------------------
The UBI searched for doesn't show coincidences, try anothe
r time!

Main:
A: Search for UBI Coincidences
B: Exit
```

Si no se encuentran coincidencias de UBI, se despliega un mensaje y se repite el ciclo.

```
Main:
A: Search for UBI Coincidences
B: Exit
A

-------------------------------------------
Enter the first 3 caracters of the UBI: 8HJT
-------------------------------------------
the UBI must be 3 caracters long, try again
-------------------------------------------
Enter the first 3 caracters of the UBI:
```

Por otro lado, si el UBI tiene más de 3 caracteres, también debes volver a ingresarlo.

```
Enter the first 3 caracters of the UBI: 9UK
-----------------------------------------
<month> <year> <M> <R>
  jan     5    0   1
  feb     5    0   0
  mar     5    0   0
  apr     5    0   0
  may     5    0   0
  jun     5    1   2
  jul     5    0   0
  aug     5    0   0
  sep     5    0   0
  oct     5    0   0
  nov     5    0   0
  dec     5    0   0

<month> <year> <M> <R>
  jan     8    0   0
  feb     8    0   0
  mar     8    0   0
  apr     8    0   1
  may     8    0   0
  jun     8    0   0
  jul     8    0   0
  aug     8    0   0
  sep     8    0   0
  oct     8    0   2
  nov     8    0   0
  dec     8    0   0
```

```
<month> <year> <M> <R>
  jan     13    0   1
  feb     13    0   1
  mar     13    0   0
  apr     13    0   0
  may     13    1   0
  jun     13    0   0
  jul     13    0   0
  aug     13    0   0
  sep     13    0   0
  oct     13    0   0
  nov     13    0   1
  dec     13    0   1

Main:
A: Search for UBI Coincidences
B: Exit
```

Si encuentra el UBI aunque sea 1 sola vez en un año, despliega todos los meses y sus respectivos contadores, después se repite el menú ciclado.

```
-------------------------------------------
Enter the first 3 caracters of the UBI: 2TO
-------------------------------------------
<month> <year> <M> <R>
  jan     15    0   0
  feb     15    0   0
  mar     15    0   0
  apr     15    0   0
  may     15    0   0
  jun     15    0   1
  jul     15    0   0
  aug     15    0   0
  sep     15    0   1
  oct     15    0   0
  nov     15    0   0
  dec     15    0   0

<month> <year> <M> <R>
  jan     18    1   0
  feb     18    0   0
  mar     18    0   0
  apr     18    0   1
  may     18    0   0
  jun     18    0   0
  jul     18    0   0
  aug     18    0   0
  sep     18    0   0
  oct     18    0   0
  nov     18    0   0
  dec     18    0   0
```

Aquí otro ejemplo de UBI encontrado.

```
Main:
A: Search for UBI Coincidences
B: Exit
B

>
```

Se termina el programa.

Miguel Jiménez Padilla A01423189

Reflexión:

Durante este bloque hemos aprendido el uso de listas ligadas y sus aplicaciones. Si bien es cierto, una lista ligada tiene muchas funciones similares a las de un arreglo común o un vector, también tiene otras ventajas, por ejemplo, poder moverte entre los elementos de la lista de una manera más sencilla mediante el uso de apuntadores, poder eliminar o agregar nodos en dónde quieras gracias a la flexibilidad de la memoria ocupada por los mismos. También, he aprendido que tienes que ser más cuidadoso a la hora de trabajar con una lista ligada porque al estar trabajando directamente con la memoria puedes causar un problema más grande a tus datos, por ello, siempre es bueno crear auxiliares en forma de apuntador para moverte por la lista. El uso de listas ligadas en esta situación problema, hizo que fuera más sencilla la creación de algunos métodos para insertar o buscar información.

Análisis de Complejidad:

Complejidad-> O(n)

```
void Lista::insertaOrden(Registro n){ // inserta un dato nuevo en orden,
recibe un numero entero
  Nodo* nuevo = new Nodo(n) ;
  if (head==NULL){//caso en que la lista está vacía
    head=nuevo;
  }
  else{
    Nodo* antes=head;
    Nodo* aux=head;
    while(aux!=NULL && aux->getDato()<n){//colocar apuntadores
      antes=aux;
      aux=aux->getSig();
    }
    if (antes==aux){ //caso insertar al inicio
      nuevo->setSig(head);
      head=nuevo;
    }
    else if (aux==NULL){//caso insertar al final
      antes->setSig(nuevo);
    }
    else{//se inserta entre dos
      antes->setSig(nuevo);
      nuevo->setSig(aux);
    }
  }
}
```

Complejidad-> O(n)

```
int Lista::buscaEnLista(string croppedUBI, int year, string month){//busca coincidencias de UBI recortado, año y mes; si encuentra coincidencias, un contador se itera y al final se regresa

  string cropped="", regMonth="";
  int regYear=0;
  int counter=0;

  if(head == NULL){ // caso si la lista esta vacia
    cout<<"The list is empty"<<endl;
   return 0;
  }

  else{ // hay elementos en la lista
    Nodo *nuevo = head;
    while(nuevo != NULL){ // while para recorrer la lista
      cropped=nuevo->getDato().getCroppedUBI();
      regMonth=nuevo->getDato().getMonth();
      regYear=nuevo->getDato().getYear();

      if(cropped==croppedUBI && regMonth==month && regYear==year){
        counter++;//se incrementa el contador
      }
      nuevo = nuevo->getSig();
    }
  }
  return counter;//regreso del contador
}
```

Complejidad-> O(n^2) (por el uso de quickSort de la librería bits/stdc++.h)

```
vector<int> sortYears(vector<int> YEARS)
{//función que regresa un vector con los años de todos los registros en el archivo de texto, ordenados y sin repeticiones
  int counter=0;
  vector<int> newYears;

  sort(YEARS.begin(),YEARS.end());
  YEARS.erase(std::unique(YEARS.begin(),YEARS.end()),YEARS.end());

  newYears=YEARS;
  return newYears;//regreso del nuevo vector
}
```

Complejidad-> O(n)

```
void ReadFile(Lista &listaM, Lista &listaR, string dataFile, vector<int> &sortedYears)
  {//función para leer los datos formateados en un archivo texto, e introducir objetos de tipo Registro en dos listas ligadas

  string day,year;
  string month, time, entry, ubi, fecha;
  vector<int> YEARS;

  ifstream archivo;
  archivo.open(dataFile);//abrir el archivo

  while(archivo>>fecha>>time>>entry>>ubi){//ciclo para acomodar los datos en las variables
    for(int o=0;o<9;o++){
      if(o<2){
        day+=fecha[o];
      }
      else if(o>2 && o<6){
        month+=fecha[o];
      }
      else if (o>6){
        year+=fecha[o];
      }
    }

    if(entry=="R"){listaR.insertaOrden(Registro(atoi(day.c_str()),month,atoi(year.c_str()),time,entry,ubi));}//caso de que entry sea R, se
    agrega en su respectiva lista

    else if(entry=="M"){listaM.insertaOrden(Registro(atoi(day.c_str()),month,atoi(year.c_str()),time,entry,ubi));}//caso de que entry sea M,
     se agrega en su respectiva lista

    YEARS.push_back(atoi(year.c_str()));//se guardan todos los años conforme se recorre el archivo

    day="";//reseteo de las variables

    month="";
    year="";
  }

  sortedYears=sortYears(YEARS);//se obtienen los años ordenados
}
```

Complejidad-> O(n)

```
void validateFiles(Lista &listaM, Lista &listaR, string dataFile, vector<int> &sortedYears)
{//Función para proporcionar el nombre del archivo en el cual se va a buscar
  while (true){
    cout<<"Enter the file name: "; cin>>dataFile;
    cout<<endl;
    ReadFile(listaM,listaR,dataFile,sortedYears);//se lee el archivo y guardan los datos en las listas

    if(listaM.longitudLista()>0 || listaR.longitudLista()>0){//romper el ciclo si el archivo tiene datos
      break;
    }
    else{//continuar el ciclo si el archivo no existe o no tiene datos
      cout<<"The file is empty or doesn't exist, try again"<<endl;
    }
  }
}
```

Complejidad-> O(n^2)

```
void UBIComparison(string &cropped, Lista &listaM, Lista &listaR, vector<int> &sortedYears )
{//función para comparar un substring de 3 caracteres con un UBI (string de un objeto Registro)

  while (true){//comprobación de que el UBI ingresado sea de 3 caracteres
    cout<<"-------------------------------------------"<<endl;
    cout<<"Enter the first 3 caracters of the UBI: ";
    cin>>cropped;
    cout<<"-------------------------------------------"<<endl;
    if(cropped.length()==3){break;}
    else{cout<<"the UBI must be 3 caracters long, try again"<<endl;}
  }

  vector<string> meses={"jan","feb","mar","apr","may","jun","jul", "aug","sep","oct","nov","dec"};

  int mCounter=0, rCounter=0, printsCOUNTER=0;//contadores de repeticiones en Mediterráneo, Rojo y de impresiones
  bool gatoControl= false,perroControl=false,aveControl=false;//variables de control

  for(int t=0; t<sortedYears.size();t++){ //ciclo que se repite por cada año en el vector
    for(int u=0;u<12;u++){//ciclo que se repite por cada mes
      //se utiliza el método busca en lista para buscar el número de ocurrencias del UBI y se iguala en su respectivo contador

      mCounter=(listaM.buscaEnLista(cropped, sortedYears[t], meses[u]));
      rCounter=(listaR.buscaEnLista(cropped, sortedYears[t], meses[u]));

      if(mCounter>0 || rCounter>0){//si alguno de los contadores es >0, significa que se va a imprimir todo ese año, entonces las variables
      de control se cambian a true y se rompe el ciclo para no trabajar de más
        gatoControl = true;perroControl=true;
        break;
      }
    }
```

```
    if(perroControl){cout<<"<month> <year> <M> <R>"<<endl;}//se imprime este encabezado

    for(int u=0;u<12;u++){//se repite el mismo ciclo para imprimir si la variable de control es true
      mCounter=(listaM.buscaEnLista(cropped, sortedYears[t], meses[u]));
      rCounter=(listaR.buscaEnLista(cropped, sortedYears[t], meses[u]));

      if (gatoControl == true){
        cout<<"  "<<meses[u]<<"     "<<sortedYears[t]<<"    "<<mCounter<<"   "<<rCounter<<endl;
        printsCOUNTER++;//se incrementa el contador de impresiones
      }
      if(u==11){aveControl=true;}
    }

    if(gatoControl){
      cout<<endl;
    }

    gatoControl = false; perroControl=false;
    mCounter=0;
    rCounter=0;

    if(t==sortedYears.size()-1 && aveControl && printsCOUNTER==0){//si es el último año y último mes y no se imprimió nada, quiere decir
    que no hubieron coincidencias de UBI
      cout<<"The UBI searched for doesn't show coincidences, try another time!"<<endl<<endl;
    }

  }
}
```