

Tecnológico Nacional de México
Campus Querétaro.

Examen parcial 3

Alumno:

Arellano Ochoa Daniel Ignacio

Carrera:

Ingeniera en sistemas computacionales

Docente:

Felipe Estrada Rojas

Fecha: 30/10/2020

Problema: Lista de preguntas

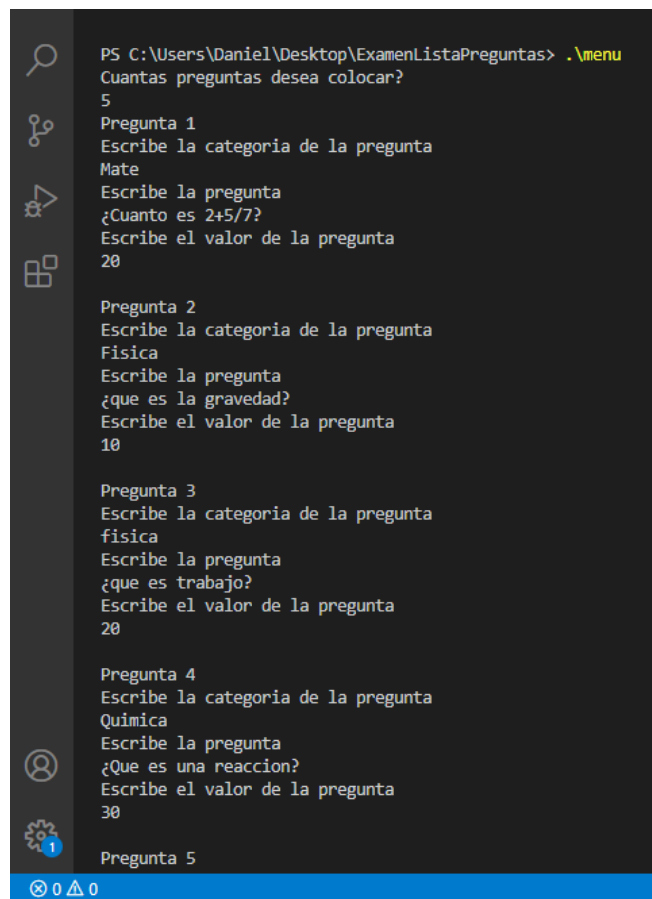
Implemente un programa utilizando estructuras lineales dinámicas que permitan controlar una lista de preguntas.

Cada pregunta deberá tener una descripción, una categoría y valor en puntos, la lista deberá estar ordenada por categorías y por puntuación, independientemente del orden como se fueron incorporando las preguntas

Al finalizar la captura de preguntas, el programa deberá proporcionar:

- Listar por categoría las preguntas capturadas
- Listar solo las categorías sin repetir (nombres diferentes de categorías)
- La cantidad de categorías y preguntas existentes
- Las preguntas de mayor a menor puntuación

Ejecución:



```
PS C:\Users\Daniel\Desktop\ExamenListaPreguntas> .\menu
Cuantas preguntas desea colocar?
5
Pregunta 1
Escribe la categoria de la pregunta
Mate
Escribe la pregunta
¿Cuanto es 2+5/7?
Escribe el valor de la pregunta
20

Pregunta 2
Escribe la categoria de la pregunta
Fisica
Escribe la pregunta
¿que es la gravedad?
Escribe el valor de la pregunta
10

Pregunta 3
Escribe la categoria de la pregunta
fisica
Escribe la pregunta
¿que es trabajo?
Escribe el valor de la pregunta
20

Pregunta 4
Escribe la categoria de la pregunta
Quimica
Escribe la pregunta
¿Que es una reaccion?
Escribe el valor de la pregunta
30

Pregunta 5
```

```
Pregunta 5
Escribe la categoria de la pregunta
mate
Escribe la pregunta
¿cual es la formula del area de un cuadrado?
Escribe el valor de la pregunta
20

Cuestionario terminado

Lista de preguntas del cuestionario:

Categoria: fisica
Pregunta 1: ¿que es trabajo? Puntuacion: 20
Pregunta 2: ¿que es la gravedad? Puntuacion: 10

Categoria: Mate
Pregunta 3: ¿Cuanto es 2+5/7? Puntuacion: 20
Pregunta 4: ¿cual es la formula del area de un cuadrado? Puntuacion: 20

Categoria: Quimica
Pregunta 5: ¿Que es una reaccion? Puntuacion: 30

Lista de categorias del cuestionario:
Categoria 1: fisica
Categoria 2: Mate
Categoria 3: Quimica

Cantidad total de preguntas: 5
Canridad total de categorias: 3

Lista de preguntas del cuestionario por puntaje:
```

```
Lista de preguntas del cuestionario:

Categoria: fisica
Pregunta 1: ¿que es trabajo? Puntuacion: 20
Pregunta 2: ¿que es la gravedad? Puntuacion: 10

Categoria: Mate
Pregunta 3: ¿Cuanto es 2+5/7? Puntuacion: 20
Pregunta 4: ¿cual es la formula del area de un cuadrado? Puntuacion: 20

Categoria: Quimica
Pregunta 5: ¿Que es una reaccion? Puntuacion: 30

Lista de categorias del cuestionario:
Categoria 1: fisica
Categoria 2: Mate
Categoria 3: Quimica

Cantidad total de preguntas: 5
Canridad total de categorias: 3

Lista de preguntas del cuestionario por puntaje:
Puntaje: 30 Pregunta 1: ¿Que es una reaccion?
Puntaje: 20 Pregunta 2: ¿Cuanto es 2+5/7?
Puntaje: 20 Pregunta 3: ¿que es trabajo?
Puntaje: 20 Pregunta 4: ¿cual es la formula del area de un cuadrado?
Puntaje: 10 Pregunta 5: ¿que es la gravedad?
Fin del programa
PS C:\Users\Daniel\Desktop\ExamenListaPreguntas>
```

Código:

menu.cpp:

```
/*
Nombre: Arellano Ochoa Daniel Ignacio
No. de control: 19141118
Problema: Lista de preguntas
Implemente un progrma utilizando estructuras lineales dinamicas que permitan
  controlar una lista de preguntas.
Cada pregunta deberá tener una descripcion, una categoria y valor en puntos,
  la lista debera estar ordenada por
  categorias y por puntuacion, independientemente del orden como se fueron inc
  orporando las preguntas
Al finalizar la captura de preguntas, el programa debera proporcionar:
    Listar por categoria las preguntas capturadas
    Listar solo las categorias sin repetir (nombres diferentes de categorias
)
    La cantidad de categorias y preguntas existentes
    Las preguntas de mayor a menor puntuacion

    He de mencionar que las categorias al ser capturas no consideran las may
    usculas por lo tanto si se escribe
    "Historia" en categoria de la primer pregunta captura y despues en otra
    pregunta se captura como categoria
    "historia" ambas seran de la misma categoria
*/
#include <string>
#include <iostream>
#include <list>
#include "Pregunta.hpp"
#include "Cuestionario.hpp"

using namespace std;

//Menu de inicio del programa
int main(int argc, const char * argv[]){
    Cuestionario* examen=new Cuestionario();//se crea el objeto de cuestiona
rio con el identificador de examen
    examen->llenarCuestionario();
    examen->listarPreguntas();
    examen->listarCategorias();
    examen->cantidadCategoriasPreguntas();
    examen->listarCuestionarioPuntuacion();
    cout<<"Fin del programa"<<endl;
}
```

Cuestionario.hpp:

```
#ifndef Cuestionario_hpp
#define Cuestionario_hpp
#include <string>
#include <iostream>
#include <list>
#include "Pregunta.hpp"
using namespace std;

class Cuestionario
{
private:
    int categorias;
    int preguntas;
    list<Pregunta*>* listaCategoria;
    list<Pregunta*>* listaPuntaje;
public:
    Cuestionario();

    void insertar(Pregunta*);
    void llenarListaPuntaje(Pregunta*);

    int compararLugar(Pregunta* , Pregunta*);
    string limpiador(string*);
    void contadorCategorias(Pregunta*);

    void llenarCuestionario();
    void listarPreguntas();
    void listarCategorias();
    void cantidadCategoriasPreguntas();
    void listarCuestionarioPuntacion();
};

#endif /*Cuestionario_hpp*/
```

Cuestionario.cpp:

```
#include "Pregunta.hpp"
#include "Cuestionario.hpp"
#include <iostream>
#include <string>
#include <list>
using namespace std;
//clase que tiene la funcion tanto de un cuestionario para ser llenado y lis
tar las preguntas o dar informacion de este
//pero a la vez tiene funciones como una lista dinamica de preguntas
Cuestionario::Cuestionario(){
    this->categorias=0;//total de categorias diferentes
    this->preguntas=0;//total de preguntas
    this-
>listaCategoria=new list<Pregunta*>();//list que almacena las preguntas acom
odadas alfabeticamente de la A a la Z segun su categoria y a la vez por su v
alor numerico de puntaje de mayor a menor
    this-
>listaPuntaje=new list<Pregunta*>();//list que almacena las preguntas acomod
ads segun su valor de puntaje de mayor a menor
}

//metodo que permite inserta un nodo de pregunta a ambas lists del cuestioan
rio, acomodando el nodo segun le corresponda
//este metodo acomoda segun la caegoria y valor numerico
void Cuestionario::insertar(Pregunta* nuevo){
    this->preguntas++;
    this-
>contadorCategorias(nuevo);//se llama otro metodo que maneja el conteo de ca
tegorias

    if(this->listaCategoria->size()==0){//caseo de estar vacia la list
        this->listaCategoria->push_front(nuevo);
    }else if(this->compararLugar(nuevo, this->listaCategoria-
>front())==1 ){//caso de insertar al inicio
        this->listaCategoria->push_front(nuevo);
    }else if(this->compararLugar(nuevo, this->listaCategoria->back())==1 || this->compararLugar(nuevo, this->listaCategoria-
>back())==0 ){//caso de insertar al final
        this->listaCategoria->push_back(nuevo);
    }else{//caso de insertar entre dos nodos
        list<Pregunta*>::iterator auxIte;//iterador que nos permita recorrer
la list
        auxIte=this->listaCategoria->begin();//desde el comienzo
```

```

        bool condicionAux=true;
        Pregunta* auxPregunta;
        while (condicionAux)
        {
            auxPregunta=*auxIte;

            if(auxIte == this->listaCategoria->end()){
                condicionAux=false;
            }

            if(this->compararLugar(nuevo, auxPregunta)==1){
                this->listaCategoria->insert(auxIte, nuevo);
                condicionAux=false;
            }else
            {
                auxIte++;
            }
        }
    }
    this->llenarListaPuntaje(nuevo); //se llama otro metodo para insertar en la lista de puntaje
}

//metodo que permite insertar un nodo en la lista acomodado segun su valor numerico
void Cuestionario::llenarListaPuntaje(Pregunta* nuevo){

    if(this->listaPuntaje->size()==0){ //en caso de estar vacia la list
        this->listaPuntaje->push_front(nuevo);
    }else if(nuevo->getValor()>this->listaPuntaje->front()->getValor()){ //en caso de insertar al inicio
        this->listaPuntaje->push_front(nuevo);
    }else if(nuevo->getValor()<=this->listaPuntaje->back()->getValor()){ //en caso de insertar al final
        this->listaPuntaje->push_back(nuevo);
    }else{
        list<Pregunta*>::iterator auxIte; //iterador que permitira recorrer la list
        auxIte=this->listaPuntaje->begin();
        bool condicionAux=true;
        Pregunta* auxPregunta;
        while (condicionAux)
        {
            auxPregunta=*auxIte;

```

```

        if(auxIte == this->listaPuntaje->end()){
            condicionAux=false;
        }

        if(nuevo->getValor()>auxPregunta->getValor()){
            this->listaPuntaje->insert(auxIte, nuevo);
            condicionAux=false;
        }else
        {
            auxIte++;
        }
    }
}
}
}

```

```

//metodo que a partir de dos nodos de pregunta indica si n1 va antes o despu
es o es igual a n2
//regresa -1 si va antes o enfrente
//regresa 1 si va despues o atras
//regresa 0 si es equivalente
//se compara primero el orden alfabetico de A a Z de la categoria y si son i
guals se compara su valor de puntaje de mayor a menor
int Cuestionario::compararLugar(Pregunta* n1, Pregunta* n2){
    int auxV1;//numeros enteros que guardaran el valor de puntaje de cada no
do de pregunta
    int auxV2;
    string a1=n1-
>getCategoria();//strings que guardaran la categoria de los nodos
    string a2=n2->getCategoria();
    string auxS1=this->limpiador(&a1);//se limpian (eliminan las mayusculas)
    string auxS2=this->limpiador(&a2);

    if(auxS1<auxS2){
        return 1;
    }else if(auxS1==auxS2){
        auxV1=n1->getValor();
        auxV2=n2->getValor();
        if(auxV1<auxV2){
            return -1;
        }else if(auxV1==auxV2){
            return 0;
        }else{

```



```

        return 1;
    }
    }else{
        return -1;
    }
}

//metodo que regresa un string sin mayusculas de esta manera la categoria "H
istoria" es lo mismo a "historia"
string Cuestionario::limpiador(string* palabra){
    char aux;//auxiliar que tomara cada dato del strign para pasarlo a minus
cula si es mayuscula
    string limpio="";//string vacio que tomara el valor del parametro pero t
odo en minusculas
    for (size_t i = 0; i < palabra->length(); i++)
    {
        aux=palabra->at(i);
        if(aux<91 && aux>64){
            aux=aux+32;
            limpio=limpio+aux;
        }else if(aux<123 && aux>96){
            limpio=limpio+aux;
        }
    }
    return limpio;
}

//metodo que ayuda a tener control del numero de categorias del cuestionario
void Cuestionario::contadorCategorias(Pregunta* nuevo){
    string categoria=nuevo-
>getCategoria();//auxiliar que tomara el valor de la categoria del nodo para
metro
    string categoriaAux="";//auxiliar que tomara el valor de una categoria

    bool condicion=true;//auxiliar que cambiara a false en caso de haber enc
ontrado en la list la misma categoria que el parametro

    list<Pregunta*>::iterator auxIte;//iterador que nos ayude a recorrer la
list
    auxIte=this->listaCategoria->begin();

    Pregunta* auxPregunta;
    while (auxIte != this->listaCategoria->end())
    {

```

```

        auxPregunta=*auxIte;

        categoriaAux=auxPregunta->getCategoria();
        if(this->limpiador(&categoria)==this->limpiador(&categoriaAux)){
            condicion=false;
            break;
        }
        auxIte++;
    }
    if(condicion){
        this->categorias++;
    }
}

//metodo que permite llenar el cuestionario
void Cuestionario::llenarCuestionario(){
    string cat, des;//auxiliares que tomaran valores del usuario para crear
    la pregunta
    int puntaje;
    Pregunta* nuevo;//pregunta que sera creada mas adelante
    int condicion;//numero de preguntas que se quieren colocar

    cout<<"Cuantas preguntas desea colocar?"<<endl;
    cin>>condicion;
    cin.ignore();

    for (size_t i = 0; i < condicion; i++)
    {
        cout<<"Pregunta " <<i+1<<endl;
        cout<<"Escribe la categoria de la pregunta"<<endl;
        getline(cin, cat);
        cout<<"Escribe la pregunta"<<endl;
        getline(cin, des);
        cout<<"Escribe el valor de la pregunta"<<endl;
        cin>>puntaje;
        cin.ignore();

        nuevo = new Pregunta(des, cat, puntaje);
        this->insertar(nuevo);
        cout<<endl;
    }
    cout<<"Cuestionario terminado"<<endl;
    cout<<endl;
    cout<<endl;
}

```

```

}

//metodo que lista todas las preguntas de la siguiente manera:
//categoria: 1:
//pregunta 1:----- Puntuacion:---
//pregunta 2:-----Puntacion:---
//categoria: 2
//pregunta 3:----Puntuacion:---
void Cuestionario::listarPreguntas(){
    string categoria="";//auxiliares que toman el valor de las categorias
    string categoriaAux="";

    list<Pregunta*>::iterator auxIte;//iterador que nos ayudara a recorrer l
a list
    auxIte=this->listaCategoria->begin();

    Pregunta* auxPregunta;//auxiliar de un nodo pregunta
    int i=1;//contador de preguntas
    cout<<"Lista de preguntas del cuestionario:"<<endl;
    while (auxIte != this->listaCategoria->end())
    {
        auxPregunta=*auxIte;

        categoriaAux=auxPregunta->getCategoria();
        if(this->limpiador(&categoria)!=this->limpiador(&categoriaAux)){
            categoria=auxPregunta->getCategoria();
            cout<<"\nCategoría: "<<categoria<<endl;
        }

        cout<<"Pregunta "<<i<<": "<<auxPregunta-
>getDescripcion()<<" Puntuación: "<<auxPregunta->getValor()<<endl;
        auxIte++;
        i++;
    }
    cout<<endl;
    cout<<endl;
}

//metodo que lista las categorias sin repetir
void Cuestionario::listarCategorias(){
    string categoria="";//auxiliares que toman el valor de categorias
    string categoriaAux="";

```

```

list<Pregunta*>::iterator auxIte;//iterador para recorrer la list
auxIte=this->listaCategoria->begin();

Pregunta* auxPregunta;//auxiliar de nod de pregunta
int i=1;          //contador de categorias
cout<<"Lista de categorias del cuestionario:"<<endl;
while (auxIte != this->listaCategoria->end())
{
    auxPregunta=*auxIte;

    categoriaAux=auxPregunta->getCategoria();
    if(this->limpiador(&categoria)!=this->limpiador(&categoriaAux)){
        categoria=auxPregunta->getCategoria();
        cout<<"Categoria "<<i<<": "<<categoria<<endl;
        i++;
    }
    auxIte++;
}
cout<<endl;
cout<<endl;
}

//metodo que imprime el total de preguntas y total de categorias
void Cuestionario::cantidadCategoriasPreguntas(){
    cout<<"Cantidad total de preguntas: "<<this->preguntas<<endl;
    cout<<"Canridad total de categorias: "<<this->categorias<<endl;
    cout<<endl;
    cout<<endl;
}

//metodo que lista las preguntas por su puntuacion de mayor a menor (no se c
onsidera su categoria)
void Cuestionario::listarCuestionarioPuntacion(){
    list<Pregunta*>::iterator auxIte;//iterador para recorrer la list
    auxIte=this->listaPuntaje->begin();

    Pregunta* auxPregunta;//auxiliar de pregunta
    int i=1;          //contador de pregunta
    cout<<"Lista de preguntas del cuestionario por puntaje:"<<endl;
    while (auxIte != this->listaPuntaje->end())
    {
        auxPregunta=*auxIte;
        cout<<"Puntaje: "<<auxPregunta-
>getValor()<<" Pregunta "<<i<<": "<<auxPregunta->getDescripcion()<<endl;
        i++;
    }
}

```

```
        auxIte++;  
    }  
}
```

Pregunta.hpp:

```
#ifndef Pregunta_hpp  
#define Pregunta_hpp  
#include <string>  
#include <iostream>  
  
using namespace std;  
class Pregunta  
{  
    private:  
        string descripcion;  
        string categoria;  
        int valor;  
        //Pregunta* ant;  
        //Pregunta* des;  
    public:  
        Pregunta(string, string, int);  
        string getDescripcion();  
        string getCategoria();  
        int getValor();  
        //Pregunta* getAnt();  
        //Pregunta* getDes();  
        //void setAnt(Pregunta*);  
        //void setDes(Pregunta*);  
};  
#endif /*Pregunta_hpp*/
```

Pregunta.cpp:

```
#include "Pregunta.hpp"
#include <string>
#include <iostream>
using namespace std;
//clase que sirve para almacenar la pregunta asi como su categoria y valor de
//puntaje, a la vez funciona como nodo
Pregunta::Pregunta(string pre, string cat, int pun){
    this->descripcion=pre;//descripcion es la pregunta que se almacenara
    this->categoria=cat;//categoria de la pregunta
    this->valor=pun;//valor de puntaje de la pregunta solo se aceptan numeros enteros
}
//metodo que regresa la categoria de la pregunta
string Pregunta::getCategoria(){
    return this->categoria;
}
//metodo que regresa la descripcion de la pregunta (regresa la pregunta almacenada)
string Pregunta::getDescripcion(){
    return this->descripcion;
}
//metodo que regresa el valor de puntaje
int Pregunta::getValor(){
    return this->valor;
}
```