CS2013 - Programación III

Simulacro Práctica Calificada #1 (PC1)

2022 - 1

Profesor: Rubén Rivas

Clases y Templates - 16 puntos

Desarrollar la clase template **pagination** que almacene una colección de números y que implemente el paginado de su contenido en una cantidad determinada de valores (**page_size**). La clase contará con un constructor con parámetros del tipo **std::initializer_list** que permita ingresar una lista de enteros al momento de crearlo.

Además, la clase debe contar con los atributos adicionales que se requieran, por ejemplo el atributo **page_size** con el valor por defecto igual a **10**, este valor define la cantidad de números que se mostrara al usar el **operador <<** sobrecargado para ostream.

La clase deberá ser implementada utilizando **arreglos dinámicos** (NO usar std::vector). Los métodos que deben implementarse son los siguientes:

- **void page_size(int value),** para actualizar el atributo **page_size.**
- **Sobrecarga del operador ++,** para avanzar a la siguiente página, si no existiera más paginas adicionales deberá quedarse en la última página.
- **Sobrecarga al operador --,** para retroceder a la anterior página, si no existiera más paginas anteriores deberá quedarse en la primera.
- **void front(),**para ubicarse en la primera página.
- **void back(),** para ubicarse en la última página.
- **int current_page (),** para retornar la página actual.
- **int pages(),** retorna la cantidad de paginas

Adicionalmente debe implementarse los constructores correspondientes para permitir copiar o mover el objeto y como se mencionó también debe implementarse la sobrecarga del **operador << ostream** de modo que al ejecutarla debe solo mostrar la cantidad de valores de la página actual (**page_size**).

```
// se ubica en la primera página y define paginamiento a 10
pagination<int> pg1 = {1, 2, 3, 4, 5, 6, 5, 10, 11, 12};
pg1.page_size(3); // cambia el tamaño de paginamiento de 10 a 3
pg1++; pg1++; pg1++; pg1++; // SOLO avanza 3 paginas
// muestra el resultado
cout << pg1 << endl;      // 12
pg1--;
cout << pg1 << endl;      // 5 10 11
pg1.front();
pg1--;
cout << pg1 << endl;      // 1 2 3
pg1.back();
cout << pg1 << endl;        // 12
pg1--; pg1--;
cout << pg1.current_page() << endl; // 2
auto pg2 = pg1;      // al copiar o mover se ubica en la página 1
pg2.page_size(4);
pg1++;
cout << pg2 << endl;     // 1 2 3 4
cout << pg1 << endl;     // 5 10 11
cout << pg2.current_page()<< " | " << pg1.current_page() << endl; // 1 | 3
```

Tema	Funcionamiento Correcto	Definición Correcta	Total
constructores, destructor	2	1	3
template de clase	2	1	3
sobrecarga operador << ostream	2	1	3
sobrecarga operadores ++, --	2	1	3
page_size, current_page, pages	1	1	2
front, back	1	1	2
			16

Templates Funciones y Contenedores - 4 puntos

Basado en la pregunta anterior generar el template de función **generate_vector** que permita copia la actual página de un objeto pagination en un vector.

```
// se ubica en la primera página y define paginamiento a 10
pagination<double> pg1 = {1, 2, 3, 4, 5, 6, 5, 10, 11, 12, 3, 4, 10};
pg1.page_size(5); // cambia el tamaño de paginamiento de 10 a 5
pg1++;
// muestra el resultado
auto vec1 = generate_vector(pg1);
for (const auto& item: vec1)
     cout << item << " ";      // 6 5 10 11 12
cout << endl;
```

Barranco, 24 de abril del 2022pa