CECS3212 Computer Programming II

C. Talavera

Programa que utiliza los operadores [] para asignar y accesar la información en un arreglo unidimensional

```
// This program demonstrates the IntegerList class.
#include <iostream>
using namespace std;
#include "IntegerList.h"
int main() {
	int  n;
	int index, value;
	cout << "Entre la cantidad de datos al arreglo:";
	cin >> n;
	IntegerList numbers(n);
	cout << "Entre los valores al arreglo:" << endl;
	for (index = 0; index < numbers.getNumElements(); index++){
		cout << "Dato[" << index + 1 << "]:";
		cin >> value;
		numbers[index] = value;
	}//end for
	cout << "El contenido del Arreglo es:";
	for (int index = 0; index < numbers.getNumElements(); index++){
		cout << numbers[value] << " ";
	}//end for
	cout << endl;

	system("pause");
	return 0;
}
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

// Specification file for the IntegerList class.
#ifndef INTEGERLIST_H
#define INTEGERLIST_H

class IntegerList
{
private:
	int *list; // Pointer to the array.
	int numElements; // Number of elements.
	bool isValid(int) const; // Validates subscripts.
public:
	IntegerList(int); // Constructor
	~IntegerList(); // Destructor
	void setElement(int, int); // Sets an element to a value.
	int getElement(int) const; // Returns an element.
	int getNumElements() const; //Retorna la cantidad de localidades del arreglo
	void subscriptError();
	int &operator[](const int &sub);
	int operator[](int subscript) const;
};
#endif

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

// Implementation file for the IntegerList class.
```

```cpp
#include <iostream>
#include <cstdlib>

using namespace std;
#include "IntegerList.h"
//***********************************************************
// The constructor sets each element to zero. *
//***********************************************************

IntegerList::IntegerList(int s)
{
       numElements = s;
       list = new int[s];

for (int ndx = 0; ndx < getNumElements(); ndx++)
        list[ndx] = 0;
}

//***********************************************************
// The destructor releases allocated memory. *
//***********************************************************

IntegerList::~IntegerList()
{
       delete [] list;
       list = nullptr;
}
//*************************************************************
// isValid member function. *
// This private member function returns true if the argument *
// is a valid subscript, or false otherwise. *
//*************************************************************

bool IntegerList::isValid(int index) const
{
       bool status;

       if (index < 0 || index >= getNumElements())
              status = false;
       else
              status = true;
       return status;
}

//************************************************************
// setElement member function. *
// Stores a value in a specific element of the list. If an *
// invalid subscript is passed, the program aborts. *
//************************************************************


void IntegerList::setElement(int index, int value){

       if (isValid(index))
              list[index] = value;

       else
       {
            cout << "Error: Invalid subscript\n";
            system("pause");
            exit(EXIT_FAILURE);
      }
}

//************************************************************
// getElement member function. *
// Returns the value stored at the specified element. *
// If an invalid subscript is passed, the program aborts. *
//************************************************************

int IntegerList::getElement(int index) const
{
      if (isValid(index)){
            return list[index];
      }
      else
      {
            cout << "Error: Invalid subscript\n";
            exit(EXIT_FAILURE);
      }
}

// Retorna la cantidad de localidades del arreglo
int IntegerList::getNumElements() const{
      return numElements;
}

 //**************************************************************
 // subscriptError function. Displays an error message and *
 // terminates the program when a subscript is out of range. *
 //**************************************************************

void IntegerList::subscriptError()
 {
 cout << "ERROR: Subscript out of range.\n";
       exit(0);
       }

 //*********************************************************
 // Overloaded [] operator. The argument is a subscript. *
 // This function returns a reference to the element *
 // in the array indexed by the subscript. *
 //*********************************************************

int &IntegerList::operator[](const int &sub)
{
       if (sub < 0 || sub >= getNumElements())
             subscriptError();
       return list[sub];
}

int IntegerList::operator[](int subscript) const
 {
       // check for subscript out-of-range error
       if (subscript < 0 || subscript >= getNumElements())
             throw out_of_range("Subscript out of range");
```

```
                return list[subscript]; // returns copy of this element
} // end function operator
```