Listas, Pilas y Colas

Estructuras de Datos Avanzadas
Universidad Nacional de San Agustín
Nombre: Alexander Rusvell Apaza Torres

1. Con Arrays
   a) Clase VStructure: Esta Clase será padre de las clases VLista, VPila y VCola.
      i. Código:

```
template <int N>

class VStructure
{
     protected:
          int array[N];
          int actual;

     public:
          VStructure(){
          this->actual=0;
          for(int i = 0; i < N; i++)
          {
          this->array[i]=0;
          }
     }

     void push(int v){
          if (this->actual<N)
          {
          this->array[actual]=v;
          this->actual++;
          }
     }

     void show(){

          cout<<"Lista: | ";
          for(int i = 0; i < N; i++)
          {
          cout<<this->array[i]<<" | ";
          }
          cout<<""<<endl;
     }

     void clear(){

          for(int i = 0; i < N; i++)
          {
          this->array[i]=0;
          }
          this->actual=0;
     }

     bool isEmpty(){
          if (actual==N)
          return false;
          return true;
     }

     bool isFull(){

          return !this->isEmpty();
     }

     ~VStructure(){

     }

};
```

b) VLista:

i. Código:

```
#include "VStructure.h"

template <int N>
class VLista : public VStructure<N>
{
    public:
        VLista(){}
        ~VLista(){}
};
```

ii. Demostración:

```
int main()
{
    VLista<5> a;
    a.show();
    a.push(1);
    a.push(2);
    a.push(3);
    a.push(4);
    a.push(5);
    a.push(4);

    cout<<endl;
    a.show();

    cout<<endl<<"Vacio: "<< a.isEmpty()<<endl;
    return 0;
}
```

```
C:\PerPer\EDA\Lab1>a.exe
Lista: | 0 | 0 | 0 | 0 | 0 |

Lista: | 1 | 2 | 3 | 4 | 5 |

Vacio: 0
```

c) VPila

i. Código

```
template <int N>
class VPila : public VStructure<N>
{
    public:
        VPila(){}
        ~VPila(){}

    int pop(){
        int res;

        this->actual--;
        res = this->array[this->actual];
        this->array[this->actual]=0;
        return res;
    }
};
```

ii. Demostración

```
int main()
{
    VPila<5> a;
    a.show();
    a.push(1);
    a.push(2);
    a.push(3);
    a.push(4);
    a.push(5);
    a.push(4);

    cout<<endl;
    a.show();
    cout<<endl<<"Vacio antes: "<< a.isEmpty()<<endl;
    cout<<a.pop()<<endl;
    a.show();
    cout<<a.pop()<<endl;
    a.show();
    cout<<a.pop()<<endl;
    a.show();
    cout<<a.pop()<<endl;
    a.show();
    cout<<a.pop()<<endl;
    a.show();

    cout<<endl<<"Vacio Despues: "<< a.isEmpty()<<endl;
    return 0;
}
```

```
C:\PerPer\EDA\Lab1>a.exe
Lista: | 0 | 0 | 0 | 0 | 0 |

Lista: | 1 | 2 | 3 | 4 | 5 |

Vacio antes: 0
5
Lista: | 1 | 2 | 3 | 4 | 0 |
4
Lista: | 1 | 2 | 3 | 0 | 0 |
3
Lista: | 1 | 2 | 0 | 0 | 0 |
2
Lista: | 1 | 0 | 0 | 0 | 0 |
1
Lista: | 0 | 0 | 0 | 0 | 0 |

Vacio Despues: 1
```

d) VCola

i. Código

```
template<int N>

class VCola : public VStructure<N>
{
	public:
		VCola(){}

		int pop()
		{
			if (this->actual==0)
				return -1;

			int res= this->array[0];
			for(int i = 1; i < this->actual ; i++){
				this->array[i-1]=this->array[i];
			}

			this->array[this->actual-1]=0;
			this->actual--;
			return res;
		}

		~VCola(){

		}
};
```

ii. Demostración

```
int main()
{
	VCola<5> a;
	a.show();
	a.push(1);
	a.push(2);
	a.push(3);
	a.push(4);
	a.push(5);
	a.push(4);

	cout<<endl;
	a.show();
	cout<<endl<<"Vacio antes: "<< a.isEmpty()<<endl;
	cout<<a.pop()<<endl;
	a.show();
	cout<<a.pop()<<endl;
	a.show();
	cout<<a.pop()<<endl;
	a.show();
	cout<<a.pop()<<endl;
	a.show();
	cout<<a.pop()<<endl;
	a.show();

	cout<<endl<<"Vacio Despues: "<< a.isEmpty()<<endl;
	return 0;
}
```

```
C:\PerPer\EDA\Lab1>a.exe
Lista: | 0 | 0 | 0 | 0 | 0 |

Lista: | 1 | 2 | 3 | 4 | 5 |

Vacio antes: 0
1
Lista: | 2 | 3 | 4 | 5 | 0 |
2
Lista: | 3 | 4 | 5 | 0 | 0 |
3
Lista: | 4 | 5 | 0 | 0 | 0 |
4
Lista: | 5 | 0 | 0 | 0 | 0 |
5
Lista: | 0 | 0 | 0 | 0 | 0 |

Vacio Despues: 1
```

2. Con Punteros

a) PStructure: Esta clase será padre de las clases PLista, PCola y PPila

i. Código

1. Clase PNodo

```
class PNodo
{
     private:
     int value;
     PNodo* siguiente;
     public:
     PNodo(int );
     ~PNodo();

     friend class PStructure;
};

PNodo::PNodo(int v)
{
     this->value=v;
     this->siguiente=0;
}

PNodo::~PNodo()
{
}
```

2. Clase PStructure

```
class PStructure
{
     protected:
          PNodo* first;
          PNodo* last;
     public:
          PStructure(){
               this->first=0;
               this->last=0;
          }
          void push(int v){
               PNodo* nuevo=new PNodo(v);
               if (last){
                    last->siguiente=nuevo;
                    last=nuevo;
                    return;
               }
               first=last=nuevo;
               return;
          }

          void show()
          {
               PNodo* temp= this->first;
               cout<<"Lista: | ";
               while(temp){
                    cout<<temp->value<<" |
                    ";
                    temp=temp->siguiente;
               }
               cout<<endl;
          }

          void remove(int position){
               PNodo* temp=this->first;
               if (position==0){
                    this->first=this->firs
                    t->siguiente;
               }

               for(int i = 0; i < position-1;
               i++){
                    temp=temp->siguiente;
               }

               cout<<endl<<"Dato Borrado:
               "<<temp->value<<endl;
               if (!temp){
                    return;
               }
               if (temp->siguiente &&
               temp->siguiente !=
               this->last){
                    temp->siguiente=temp->
                    siguiente->siguiente;
               }
               else if(temp->siguiente &&
               temp->siguiente==last){
                    temp->siguiente=0;
               }
               else if(!temp->siguiente){
                    this->first=0;
                    this->last=0;
               }
          }

          ~PStructure(){}
};
```

b) Clase PLista

i. Código:

```
class PLista : public PStructure
{
    public:
        PLista(){}
        ~PLista(){}
};
```

ii. Demostración:

```
int main()
{
    PLista a;
    a.push(1);
    a.push(2);
    a.push(3);
    a.push(4);
    a.push(5);
    a.push(6);
    a.show();
    a.remove(0);
    a.show();
    a.remove(0);
    a.show();
}
```

```
C:\PerPer\EDA\Lab1>a.exe
Lista: | 1 | 2 | 3 | 4 | 5 | 6 |

Dato Borrado: 1
Lista: | 2 | 3 | 4 | 5 | 6 |

Dato Borrado: 2
Lista: | 3 | 4 | 5 | 6 |
```

c) Clase PCola

i. Código

```
class PCola : public PStructure
{
    public:
        PCola(){}
        int pop()
        {
            PNodo* temp=this->first;
            PStructure::remove(0);
            return temp->value;
        }

        ~PCola(){}
};
```

ii. Demostración

```
int main()
{
    PCola a;
    a.push(1);
    a.push(2);
    a.push(3);
    a.push(4);
    a.push(5);
    a.push(6);
    a.show();
    a.pop();
    a.show();
    a.pop();
    a.show();
    a.pop();
    a.show();
}
```

```
C:\PerPer\EDA\Lab1>a.exe
Lista: | 1 | 2 | 3 | 4 | 5 | 6 |

Dato Borrado: 1
Lista: | 2 | 3 | 4 | 5 | 6 |

Dato Borrado: 2
Lista: | 3 | 4 | 5 | 6 |

Dato Borrado: 3
Lista: | 4 | 5 | 6 |
```

d) Clase PPila

i. Código

```
class PPila : public PStructure
{
    public:
        PPila(){}
        int pop()
        {
            PNodo* temp= this->first;
            while(temp->siguiente!= this->last){
                temp=temp->siguiente;
            }
            int res=temp->siguiente->value;
            temp->siguiente=0;
            this->last=temp;
            cout<<endl<<"Dato Borrado: "<<res<<endl;
            return res;
        }
        ~PPila(){}
};
```

ii. Demostración

```
int main()
{
    PPila a;
    a.push(1);
    a.push(2);
    a.push(3);
    a.push(4);
    a.push(5);
    a.push(6);
    a.show();
    a.pop();
    a.show();
    a.pop();
    a.show();
    a.pop();
    a.show();
}
```

```
C:\PerPer\EDA\Lab1>a.exe
Lista: | 1 | 2 | 3 | 4 | 5 | 6 |

Dato Borrado: 6
Lista: | 1 | 2 | 3 | 4 | 5 |

Dato Borrado: 5
Lista: | 1 | 2 | 3 | 4 |

Dato Borrado: 4
Lista: | 1 | 2 | 3 |
```