Practica 1: Clases

1.1. Introduccion básica

```
//////////////////////////////////////////////////
main.cpp
//////////////////////////////////////////////////

#include <integer.h>
#include <iostream>
using namespace std;
using namespace values;
int main(int argc,char **argv)
{

	Integer I;

	cout<<I.getValue()<<endl;
	I.setValue(10);

	cout<<I.getValue()<<endl;

};
```

```
//////////////////////////////////////////////////
makefile
//////////////////////////////////////////////////

all:create

create:
	g++ -I. main.cpp integer.cpp -o integer
clean:
	rm *.o *~ integer -f
```

```
//////////////////////////////////////////////////
integer.h
//////////////////////////////////////////////////
#ifndef _Integer_h_
#define _Integer_h_
namespace values
{

class Integer
{
public:

	/** Returns the current value of the integer
```

```
        */
        int getValue();

        /**Sets a value in this object
        */
        void setValue(int v);


private:
int _ivalue;
};

}

#endif
```

```
//////////////////////////////////////////////////
integer.cpp
//////////////////////////////////////////////////
#include <integer.h>
namespace values{

////////////////////////////////////
//
////////////////////////////////////
int Integer::getValue()
{
        return _ivalue;
}

////////////////////////////////////
//
////////////////////////////////////
void Integer::setValue(int v)
{
        _ivalue=v;
}

}
```

1.2 Constructores, y sobrecarga de constructores

```
//////////////////////////////////////////////////
integer.h
```

```
////////////////////////////////////////////////
#ifndef _Integer_h_
#define _Integer_h_
namespace values
{

class Integer
{
public:
        /**Empty constructor
        */
        Integer();

        /**Parametrized constructor
        */
        Integer(int v);
        /**Copy constructor
        */
        Integer(const Integer &I);

        /** Returns the current value of the integer
        */
        int getValue();

        /**Sets a value in this object
        */
        void setValue(int v);


private:
int _ivalue;
};

}

#endif
```

```
////////////////////////////////////////////////////
integer.cpp
////////////////////////////////////////////////////
#include <integer.h>
namespace values{

//////////////////////////
//
//////////////////////////
Integer::Integer()
{
_ivalue=0;
}
//////////////////////////
//
```

```
//////////////////////
Integer::Integer(int v)
{
_ivalue=v;

}

//////////////////////////
//
//////////////////////
Integer::Integer(const Integer & I)
{
_ivalue=I._ivalue;
}
.
.
.
.
.
.
}
```

```
/////////////////////////////////////////////////
main.cpp
/////////////////////////////////////////////////

#include <integer.h>
#include <iostream>
using namespace std;

int main(int argc,char **argv)
{

        values::Integer I1(10);

        cout<<I1.getValue()<<endl;
        values::Integer I2(I1);
        cout<<I2.getValue()<<endl;


};
```

1.3 Operadores

```
/////////////////////////////////////////////////
integer.h
/////////////////////////////////////////////////
#ifndef _Integer_h_
#define _Integer_h_
namespace values
{
```

```
class Integer
{
public:
	/**Empty constructor
	*/
	Integer();

	/**Parametrized constructor
	*/
	Integer(int v);
	/**Copy constructor
	*/
	Integer(const Integer &I);

	/**Copy operator
	*/
	Integer & operator=(const Integer & I);

	/** Returns the current value of the integer
	*/
	int getValue();

	/**Sets a value in this object
	*/
	void setValue(int v);


private:
int _ivalue;
};

}

#endif


////////////////////////////////
integer.cpp
//////////////////
.
.
.
Integer & Integer::operator=(const Integer & I)
{
	_ivalue=I._ivalue;
	return *this;
}



.
.
.
```

```
//////////////////////////////////////////////////
main.cpp
//////////////////////////////////////////////////

#include <integer.h>
#include <iostream>
using namespace std;

int main(int argc,char **argv)
{

	values::Integer I1(10);

	cout<<I1.getValue()<<endl;
	values::Integer I2=I1;
	cout<<I2.getValue()<<endl;


};
```