

Actividad 1.2: Clases Conceptos Básicos

Variable Name	Description
Shares	Variable tipo int que almacena los shares que el usuario quiere
price	Variable tipo double que almacena el precio de los stocks
Commission	Variable tipo double que almacena la commission del broker
Member Function	Description
Stock();	Constructor por defecto, se inicializa con 0
Stock(int aShares);	Constructor con un parámetros de tipo int, que se le asigna una variable.
setShares	Función que almacena los shares del tipo entero
getShares	Funcion que retorna los el valor de los shares del usuario
getprice	Funcion que retorna la variable price
getcommission	Funcion que retorna la variable Commission
display	Funcion que imprime los resultados.
~Stock	Funcion destructora

Stock
- int shares - double price - double Commission
+Stock(); +Stock(int aShares); +void setShares(int aShares); +int getShares()const; +double getprice()const; +double getcommission()const; +void display()const; +~Stock();

Stock.h

```
#ifndef Stock_H
```

```
#define Stock_H
```

```
class Stock {
```

```
private:
```

```
int shares;

double price = 35.00;

double Commission = 0.02;

public:

Stock();

Stock(int aShares);

void setShares(int aShares);

int getShares()const;

double getprice()const;

double getcommission()const;

void display()const;

~Stock();// destructora

};

#endif
```

Stock.cpp

```
#include <iostream>

#include "Stock.h"

using namespace std;

Stock::Stock() {

setShares(0);

}

Stock::Stock(int aShares) {

setShares(aShares);
```

```

}

void Stock::setShares(int aShares) {
    shares = aShares;
}

int Stock::getShares()const {
    return shares;
}

double Stock::getprice()const {
    return price;
}

double Stock::getcommission()const {
    return Commission;
}

void Stock::display()const {
    cout << "The amount paid for the stock alone (without the commission): $" <<
    getShares() * getprice() << endl;

    cout << "The amount of the commission: $" << getcommission() * (getprice() *
    getShares()) << endl;

    cout << "The total amount paid (for the stock plus the commission): $" <<
    getcommission() * (getprice() * getShares()) + (getShares() * getprice()) << endl;
}

Stock::~~Stock() {
    cout << "*Execute destruction*\n";
}

```

StockMain.cpp

```
#include <iostream>
```

```
#include "Stock.h"
```

```
using namespace std;
```

```
int main() {
```

```
int shares;
```

```
Stock p1;
```

```
cout << "Enter how many shares of stock u want: ";
```

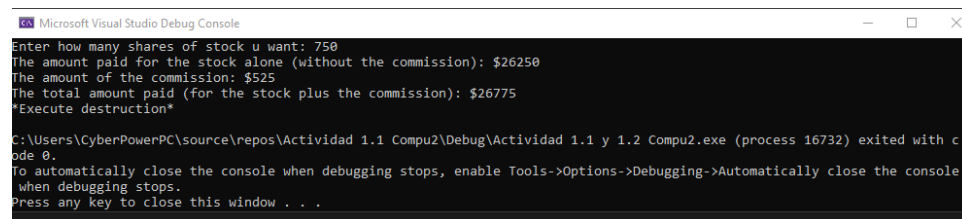
```
cin >> shares;
```

```
p1.setShares(shares);
```

```
p1.display();
```

```
}
```

Output

A screenshot of the Microsoft Visual Studio Debug Console window. The window title is "Microsoft Visual Studio Debug Console". The output text is as follows:

```
Enter how many shares of stock u want: 750
The amount paid for the stock alone (without the commission): $26250
The amount of the commission: $525
The total amount paid (for the stock plus the commission): $26775
*Execute destruction*

C:\Users\CyberPowerPC\source\repos\Actividad 1.1 Compu2\Debug\Actividad 1.1 y 1.2 Compu2.exe (process 16732) exited with c
ode 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console
when debugging stops.
Press any key to close this window . . .
```

Enter how many shares of stock u want: 750

The amount paid for the stock alone (without the commission): \$26250

The amount of the commission: \$525

The total amount paid (for the stock plus the commission): \$26775

Execute destruction

C:\Users\CyberPowerPC\source\repos\Actividad 1.1 Compu2\Debug\Actividad 1.1 y 1.2 Compu2.exe
(process 16732) exited with code 0.

To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.

Press any key to close this window . . .