FACULTAD DE INFORMÁTICA

El lenguaje C++

Una introducción para programadores

SOLUCIONES DEL TALLER

Programación orientada a objetos — Unidad 0

Autor: Luis Hernández Yáñez

Un ejercicio sobre sintaxis (*resolución individual*)

El siguiente programa contiene bastantes errores de sintaxis. Intenta detectar el mayor número de errores que puedas (señala dónde se encuentra la causa). Aunque el compilador te ayudaría mucho a encontrar los errores, intenta encontrarlos por ti mismo.

```
#include <iostream>
using namesspace std;
#define MAX 100

char menu();
boolean divisivol(int; int);
void prueba_for();
void prueba_while();

int main() {
  cout << "Bienvenido al programa de prueba\n";
  char c =< menu();
```



```
  switch(c) {
    case '1':
      prueba_for();
      break;
    case '2':
      prueba_while();
      break;
    case '3':
      integer num1, num2;
      cout << "Dame un entero: ";
      cin >> num1;
      cout << "Dame otro: ";
      cin >> num2;
      if(divisible(num1, num2)) then
        cout << "El primero es divisible por el segundo\n";
      else
        cout << "El primero no es divisible por el primero\n";
      break;
  }
  return 0;
};
```



```
char menu() {
  char c;
  do {
    cout << "Lo que podemos mostrarte:\n";
    cout << "1 - Prueba del for\n";
    cout << "2 - Prueba del while\n";
    cout << "3 - Prueba de funcion\n";
    cout << "Quieres... (1 a 3): ";
    cin >> c;
    if((c < '1') or (c > '3')) cout << "Intenta otra vez...\n";
  } while((c < '1') or (c > '3'));
  return c;
}

bool divisible(int numero1, int numero2) {
  if(numero1 mod numero2 == 0) return true;
  return false;
}
```

La sintaxis del lenguaje C++

```
void prueba_for() {
  cout << "Uso un for para repetir algo...\n";
  for(int i := 1, i <=MAX, i++) cout >> (i % 10);
  cout << "\n";
}

void prueba_while() {
  cout << "Mientras que 'letra' no sea la 'x' yo sigo...\n";
  char letra = 'a';
  while(letra <> "x") {
    cout << "Dame "letra": ";
    cin >> letra;
  };
}
```

Y, de paso, ¿has entendido lo que hace el programa?

Cuestiones

Sobre las diferencias entre Pascal y C++ (*resolución individual*)

¿Qué mecanismos has visto en C++ que no tengan equivalente en Pascal?

Las bibliotecas (módulos)
Los modificadores de tipos
Los especificadores de clase de almacenamiento
Los operadores de incremento y decremento
Los moldes
La inicialización dinámica
Las abreviaturas
`break` y **`continue`**
Prototipos
Argumentos implícitos
Sobrecarga de operadores y funciones
El preprocesador

Cuestiones

Sobre las diferencias entre Pascal y C++ (*resolución individual*)

De las construcciones de bucles de C++, ¿has visto alguna que no sea exactamente equivalente a las que hay en Pascal?

El **`do...while`** (su sentido es el contrario)

Respecto de lo que conoces del lenguaje Pascal, ¿hay algo que eches de menos en lo que hemos visto del lenguaje C++?

Arrays y cadenas de caracteres
¿Punteros?
Archivos (ficheros)
Conjuntos

Ejercicio práctico

Ejercicio práctico sobre el lenguaje C++ (*resolución en grupo*)

Se trata de desarrollar un TAD (Tipo abstracto de datos) **`Complejo`** que permita trabajar con números complejos en los programas.

Usaremos sólo lo que hemos visto hasta ahora de C++.

La funcionalidad que se desea para el TAD es la básica:

- ✓ Construcción de un complejo a partir de sus partes real e imaginaria
- ✓ Suma de complejos
- ✓ Resta de complejos
- ✓ Multiplicación de complejos
- ✓ División de complejos
- ✓ Visualización de complejos en una forma del tipo **`(2 + 4i)`**

Ejercicio práctico

Pasos

1. Lo primero que tenéis que hacer es centraros en los datos necesarios.
2. Una vez que hayáis establecido la estructura de datos adecuada, deberéis pasar a implementar las seis operaciones.
3. Superado el paso anterior deberéis convertir el código en una biblioteca, de forma que esté disponible el TAD para cualquier programador.
4. Desarrollad un sencillo programa principal que use todas las características del TAD **`Complejo`**.

Otros requisitos

Para implementar las operaciones aritméticas utilizad funciones operadoras.

Utiliza comentarios donde te parezcan útiles (*piensa en los demás*).

Ejercicio práctico

1.- Los datos

```
struct Complejo {
  double real, imag;
};
```


Ejercicio práctico

2.- Implementación de las operaciones

```
struct Complejo {
  double real, imag;
};

Complejo Construye(double, double);
Complejo operator+(Complejo, Complejo);
Complejo operator-(Complejo, Complejo);
Complejo operator*(Complejo, Complejo);
Complejo operator/(Complejo, Complejo);
void mostrar(Complejo);

Complejo Construye(double r, double i) {
  Complejo comp;
  comp.real = r;
  comp.imag = i;
  return comp;
}
```

(*continúa*)

Ejercicio práctico

```
Complejo operator+(Complejo c1, Complejo c2) {
  Complejo comp;
  comp.real = c1.real + c2.real;
  comp.imag = c1.imag + c2.imag;
  return comp;
}

Complejo operator-(Complejo c1, Complejo c2) {
  Complejo comp;
  comp.real = c1.real - c2.real;
  comp.imag = c1.imag - c2.imag;
  return comp;
}

Complejo operator*(Complejo c1, Complejo c2) {
  Complejo comp;
  comp.real = c1.real * c2.real - c1.imag * c2.imag;
  comp.imag = c1.real * c2.imag + c1.imag * c2.real;
  return comp;
}
```

(*continúa*)

Ejercicio práctico

```
Complejo operator/(Complejo c1, Complejo c2) {
  Complejo comp;
  // Hacemos que comp sea el complementario de c2
  comp.real = c2.real;
  comp.imag = - c2.imag;
  // Multiplicamos c1 por el complementario de c2 (comp)
  comp = c1 * comp;
  // Sólo queda dividir por la suma de los cuadrados
  // de las partes real e imaginaria de c2
  double divisor = c2.real * c2.real + c2.imag * c2.imag;
  comp.real = comp.real / divisor;
  comp.imag = comp.imag / divisor;
  return comp;
}

void mostrar(Complejo c) {
  cout << "(" << c.real << " + " << c.imag << "i)\n";
}
```


Ejercicio práctico

3.- Conversión en una biblioteca

Archivo **complejo.h**

```
#include <iostream>
using namespace std;

struct Complejo {
  double real, imag;
};

Complejo Construye(double, double);
Complejo operator+(Complejo, Complejo);
Complejo operator-(Complejo, Complejo);
Complejo operator*(Complejo, Complejo);
Complejo operator/(Complejo, Complejo);
void mostrar(Complejo);
```


Ejercicio práctico

Archivo **complejo.cpp**

```
#include "complejo.h"

Complejo Construye(double r, double i) {
  Complejo comp;
  comp.real = r;
  comp.imag = i;
  return comp;
}

Complejo operator+(Complejo c1, Complejo c2) {
  Complejo comp;
  comp.real = c1.real + c2.real;
  comp.imag = c1.imag + c2.imag;
  return comp;
}

// Resto de funciones
```


Ejercicio práctico

4.- Un programa principal que use el TAD

```
#include "complejo.h"

int main() {
  Complejo comp1, comp2, comp3;
  comp1 = Construye(12,3);
  mostrar(comp1);
  comp2 = Construye(3,4);
  mostrar(comp2);
  comp3 = comp1 + comp2;
  mostrar(comp3);
  comp3 = comp1 - comp2;
  mostrar(comp3);
  comp3 = comp1 * comp2;
  mostrar(comp3);
  comp3 = comp1 / comp2;
  mostrar(comp3);
  return 0;
}
```