Unidad Nº3

Preguntas orientadoras

1) ¿Qué significa el modificador *const* situado a la derecha de los paréntesis, en la declaración de un método?

2) ¿ Cómo se declaran dos clases mutuamente amigas?

3) ¿Qué diferencia hay entre un constructor de copia y el operador de asignación (=)?

4) ¿Cuándo se llama al constructor de copia?

5) Al sobrecargar funciones miembro de una clase, ¿de qué manera deben diferir?

6) ¿Por qué utilizar valores predeterminados si se puede sobrecargar una función?

7) ¿Qué operadores no se pueden sobrecargar en C++?

8) ¿Para qué sirve el operador const_cast<type>?

Ejercicios

1) Escribir un programa en C++ que permite realizar operaciones con números complejos: suma, resta, multiplicación y división. Además, a cualquier número complejo se le puede pedir su módulo y ángulo. Este último deberá pertenecer al intervalo [0, 2pi].

2) Observe el programa que sigue y piense cual será su salida. Luego, escriba el código en su IDE favorito, compile y ejecute para verificar lo que usted pensó.

```cpp
#include <string>
#include <iostream>
using namespace std;
int main()
{
    string bigNews("I saw Elvis in a UFO. ");
    cout << bigNews << endl;
    // How much data have we actually got?
    cout << "Size = " << bigNews.size() << endl;
    // How much can we store without reallocating?
    cout << "Capacity = " << bigNews.capacity() << endl;
    // Insert this string in bigNews immediately
    // before bigNews[1]:
    bigNews.insert(1, " thought I");
    cout << bigNews << endl;
    cout << "Size = " << bigNews.size() << endl;
    cout << "Capacity = " << bigNews.capacity() << endl;
    // Make sure that there will be this much space
    bigNews.reserve(500);
    // Add this to the end of the string:
    bigNews.append("I've been working too hard.");
    cout << bigNews << endl;
    cout << "Size = " << bigNews.size() << endl;
    cout << "Capacity = " << bigNews.capacity() << endl;
}
```

3) El siguiente programa usa las funciones *insert*, *find* y *replace* del objeto string. Estudie las sentencias *assert* y verifique su funcionamiento. Piense las diferencias entre insertar y reemplazar una cadena en otra. ¿Cómo trabaja la función *replace*?

```cpp
#include <cassert>
#include <string>
using namespace std;
int main()
{
    string s("A piece of text");
    string tag("$tag$");
    s.insert(8, tag + ' ');
    assert(s == "A piece $tag$ of text");
    int start = s.find(tag);
    assert(start == 8);
    assert(tag.size() == 5);
    s.replace(start, tag.size(), "hello there");
    assert(s == "A piece hello there of text");
}
```

4) Modifique el programa del ejercicio 3 para poder reemplazar un string ingresado por teclado dentro del string s. Se debe controlar que la cadena ingresada por teclado exista dentro de la cadena s. ¿Qué sucede con el tamaño

del string s si la cadena a reemplazar tiene un tamaño igual o menor que el de la cadena reemplazada? ¿ Aumenta, disminuye o se mantiene el tamaño de s? ¿Se le ocurre un caso en el que el reemplazo haga que el tamaño de s pueda aumentar?

5) El siguiente listado es un programa de prueba llamado test.cpp para probar el funcionamiento de una clase denominada CRacional. Copie el código en su IDE favorito y a continuación, haga que compile sin advertencias y ejecute.

```
#include <iostream>
#include "racional.h"
using namespace std;

int main()
{
 CRacional a, b(3, 7), c;
 int d = 3;

 cout << "a: "; cin >> a;
 c = a + b;

 c += b;
 cout << c << endl;
 c = a + b; // equivale a: c = a.operator+(b)
 cout << c << endl;
 c = b + a; // equivale a: c = b.operator+(a)
 cout << c << endl;
 c = a + static_cast<CRacional>(d);
 cout << c << endl;
 c = static_cast<CRacional>(d) + a;
 cout << c << endl;

 double x = 2.0;
 c = x + a;
 cout << c << endl;

 if (a == b) cout << "a es igual a b\n";
 if (a < b) cout << "a es menor que b\n";
 if (a > b) cout << "a es mayor que b\n";
```

```
    if (!a) cout << "racional nulo\n";

    c = ++a;
    cout << "c = " << c << endl;
    cout << "a = " << a << endl;
    c = a++;
    cout << "c = " << c << endl;
    cout << "a = " << a << endl;

    c = --a;
    c = a--;

    c = -a + b;
    cout << c << endl;

    c = -a - b;
    cout << c << endl;

    c = a * b;
    cout << c << endl;

    c = a / b;
    cout << c << endl;

    x = c;
    cout << x << endl;

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

6) La imagen que sigue muestra la declaración de una clase string básica. Se pide el código que implemente esa clase y un programa de prueba que demuestre su funcionamiento.

```
#if !defined(_CADENA_H_)
#define _CADENA_H_

#include <iostream>
using namespace std;

//////////////////////////////////////////////////////////////////////
// Clase para operar con cadenas de caracteres
class CCadena
{

  private:
    char   *pmCad;    // puntero a la cadena de caracteres
    size_t nlong;     // longitud de la cadena

  public:
    CCadena(const char * = 0);          // constructor
    CCadena(const CCadena&);            // constructor copia
    CCadena(char, int);                 // constructor
    ~CCadena();                         // destructor

    // Concatenar cadenas de caracteres
    friend CCadena operator+(const CCadena&, const CCadena&);

    // Comparación de cadenas
    friend bool operator==(const CCadena&, const CCadena&);
    friend bool operator!=(const CCadena&, const CCadena&);
    friend bool operator< (const CCadena&, const CCadena&);
    friend bool operator> (const CCadena&, const CCadena&);

    // Operaciones de entrada/salida
    friend istream& operator>>(istream&, CCadena&);
    friend ostream& operator<<(ostream&, const CCadena&);

    // Asignación, concatenación e indexación
    CCadena& operator=(const CCadena&); // asignación objeto
    CCadena& operator=(const char *);   // asignación cadena
    CCadena operator+=(const CCadena&); // suma mas asignación
    char& operator[](unsigned int);     // indexación
    size_t ObtenerLong() const { return nlong; }
};

#endif // _CADENA_H
//////////////////////////////////////////////////////////////////////
```