

95.12 Algoritmos y Programación II

Práctica 1: C++ básico

Notas preliminares

- Esta práctica tiene como meta ejercitar los conceptos introductorios de C++, que es el lenguaje de programación que utilizaremos en la materia.
- Los ejercicios marcados con el símbolo ♣ constituyen un subconjunto mínimo de ejercitación. No obstante, recomendamos fuertemente realizar todos los ejercicios.

Ejercicio 1

Escribir un programa que imprima los tamaños de los tipos fundamentales y de varios tipos de punteros, usando el operador `sizeof`.

Ejercicio 2

Verificar si el compilador genera código equivalente para iterar usando punteros,

```
for (char *p = v; *p; ++p)
    usar(*p);
```

y usando índices,

```
for (int i = 0; v[i]; ++i)
    usar(v[i]);
```

Analizar informalmente cómo influye el nivel de optimización del compilador en la calidad del código generado.

Ejercicio 3

Escribir un programa que lea caracteres del stream estándar de entrada, *cin*, y escriba la información codificada por *cout*. La forma de encriptar un carácter *d* es,

```
e = d ^ key[i];
```

donde *key* es un string pasado por línea de comando. De ser necesario, el programa deberá usar los caracteres de *key* en forma cíclica (notar que re-encriptar texto con la misma llave produce el texto original).

Ejercicio 4 ♣

Explicar las diferencias entre:

- `const tipo &r`
- `tipo const &r`
- `tipo& const r`
- `const tipo *p`
- `tipo const *p`

- tipo* const p
- const tipo* const p

Ejercicio 5 ♣

Explicar, desde el punto de vista de C++, las diferencias entre estructuras y clases.

Ejercicio 6 ♣

Implementar una clase arreglo, con las operaciones necesarias para poder ejecutar este programa:

```
int main()
{
    arreglo a(7);

    a[0] = 3;
    a[1] = 6;
    a[2] = a[0] + a[1];
    arreglo b = a;
    std::cout << b[2] << std::endl;
}
```

Ejercicio 7

- (a) Implementar una clase complejo, definiendo adecuadamente los operadores aritméticos usuales. Escribir un programa que reciba como entrada una secuencia $x_0 \dots x_{n-1}$ de números complejos y calcule e imprima,

$$X_k = \sum_{i=0}^{n-1} x_i \cdot e^{-j2\pi \frac{ki}{n}}; 0 \leq k \leq n-1$$

- (b) Reescribir la clase complejo y usar notación polar para la representación interna de los números. Verificar que, a igual entrada, ambos programas producen salidas similares.

Ejercicio 8

Explicar la salida de este programa:

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

class foo {
public:
    foo() { cout << "foo::foo()" << endl; }
    foo(const foo&) { cout << "foo::foo(const foo&)" << endl; }
    ~foo() { cout << "foo::~foo()" << endl; }
};

foo bar(foo A)
{
    cout << "foo bar(foo)" << endl;
    return A;
}
```

```
int main()
{
    foo A;
    bar(A);
}
```

Ejercicio 9 ♣

Explicar la salida de este programa:

```
class foo {
public:
    foo() { cout << "foo::foo()" << endl; }
    foo(const foo &) { cout << "foo::foo(const foo&)" << endl; }
    foo const &operator=(foo const &f) {
        cout << "foo const &operator=(foo const &)" << endl;
        return *this;
    }
    ~foo() { cout << "foo::~~foo()" << endl; }
};

foo bar(foo A)
{
    cout << "foo bar(foo)" << endl;
    return A;
}

int main()
{
    foo A;
    foo B(A);
    foo C = A;
    A = bar(B);
}
```

Ejercicio 10 ♣

Proponer una clase para operar con aritmética racional; y con al menos dos atributos enteros: numerador y denominador, que representan el número en su forma reducida (es decir, estos números son coprimos; y el cero se representa como 0/1).

Implementar la operaciones de suma, resta, producto, cociente, entrada/salida con formato y conversión a los tipos escalares nativos como float y double.