Trabajo práctico 3: Implementación
CalculadoraProgramable

Normativa

Límite de entrega: Presencialmente, cualquiera de los siguientes miércoles:

- Miércoles 27 de junio.
- Miércoles 4 de julio.
- Miércoles 11 de julio.

Y digitalmente enviando el zip al mail `algo2.dc+tp3@gmail.com`. Ver detalles de entrega abajo.

Normas de entrega: Ver "Información sobre la cursada" en el sitio Web de la materia.
(`http://campus.exactas.uba.ar`)

1. Enunciado

Este TP consiste en implementar en C++ todos los módulos correspondientes a la calculadora programable diseñada en el TP 2. Esto incluye la clase `Calculadora` y todos los módulos auxiliares necesarios para su funcionamiento, incluyendo la clase `Programa`, y todos los diccionarios e iteradores que hayan diseñado.

El código debería respetar el diseño elegido en el TP 2 de la manera más fiel posible. Obviamente se permite y se espera que corrijan todos los potenciales *bugs* que puedan llegar a encontrar en el diseño, pero cualquier cambio en las estructuras o algoritmos tiene que estar debidamente justificado.

1.1. Módulos básicos

Pueden utilizar las siguientes clases de la STL para los respectivos módulos básicos:

Módulo	**Clase**
Lista Enlazada	`std::list`
Pila	`std::stack`
Cola	`std::queue`
Vector	`std::vector`
Diccionario Lineal	`std::map`
Conjunto Lineal	`std::set`

Además, se provee una implementación de la cátedra del módulo Ventana en los archivos `Ventana.h` y `Ventana.hpp`.

1.2. Detalles de implementación

Como es usual en los trabajos del laboratorio, el proyecto debe estar escrito utilizando alguna versión del lenguaje `C++` que se encuentre soportada por el compilador `g++` (por ejemplo C++11 o C++14).

En el directorio `src/` se deben ubicar todos los `.h`, `.cpp` y `.hpp`. Los distintos tipos de archivo deben respetar su finalidad esperada, reservando los `.h` para las declaraciones de clases y funciones, los `.cpp` para la implementación de los métodos de las clases que no sean genéricas, y los `.hpp` para la implementación de los métodos de las clases genéricas (declaradas con `template`).

Los tests se deben ubicar en el directorio `tests/` y deben estar escritos utilizando el entorno Google Test.

El proyecto debe incluir un archivo `CMakeLists.txt` que permita compilarlo con `cmake`.

1.3. Testing

1.4. Tests de unidad

Cada clase implementada debe venir acompañada de un conjunto de tests en un archivo separado que compruebe su correcta funcionalidad. Por ejemplo, la clase `Calculadora` debe incluir un conjunto de tests en el archivo `tests/test_calculadora.cpp`. No se pide que los tests sean exhaustivos, pero sí razonablemente completos y representativos.

1.5. Driver de test

Además, el TP debe pasar el conjunto de tests diseñados por la cátedra, que se encuentran en el archivo `tests/test_driver.cpp`. Puesto que cada grupo puede utilizar una interfaz diferente para las clases `Programa` y `Calculadora`, dependiendo del diseño que cada grupo haya elegido en el TP 2, los tests interactúan con la calculadora exclusivamente a través de la clase `Driver`.

La clase `Driver` es una "fachada" o *wrapper* cuyo objetivo es poder crear un programa y ejecutarlo en una calculadora, tal como se detalla a continuación. Para utilizar correctamente el driver de test:

- **No deben** modificar los tests en `tests/test_driver.cpp`.
- **Deben** completar la parte privada y la implementación de los métodos de la clase `Driver` en `src/Driver.h` y `src/Driver.cpp`.

La clase `Driver` cuenta con **un único** programa que se inicializa vacío. El `Driver` opera en dos etapas. Durante la primera etapa, el usuario puede agregar rutinas e instrucciones al programa. Durante la segunda etapa, el usuario puede ejecutar ese único programa en una calculadora, también única. Más precisamente, los métodos de la clase `Driver` son:

Métodos para construir el programa.

- `void Driver::begin(string rutina)` — Marca el comienzo de una rutina.
- `void Driver::end(string rutina)` — Marca el final de una rutina.
- `void Driver::push(int n)` — Agrega una instrucción PUSH(n) a la rutina actual.
- `void Driver::add()` — Agrega una instrucción ADD a la rutina actual.
- `void Driver::sub()` — Agrega una instrucción SUB a la rutina actual.
- `void Driver::mul()` — Agrega una instrucción MUL a la rutina actual.
- `void Driver::read(string x)` — Agrega una instrucción READ(x) a la rutina actual.
- `void Driver::write(string x)` — Agrega una instrucción WRITE(x) a la rutina actual.
- `void Driver::jump(string r)` — Agrega una instrucción JUMP(r) a la rutina actual.
- `void Driver::jumpz(string r)` — Agrega una instrucción JUMPZ(r) a la rutina actual.

Métodos para ejecutar el programa

- `void Driver::comenzarEjecucion(string rutina, int capacidadVentana)`
  Comienza la segunda etapa. Construye una nueva calculadora con la capacidad de ventana indicada para ejecutar la rutina indicada del programa ya construido. Una vez iniciada la segunda etapa, no se deben agregar más rutinas ni instrucciones al programa, ni se debe volver a invocar al método `comenzarEjecucion`.
- `void Driver::asignarVariable(string x, int valor)`
  Asigna el valor dado a la variable x.
- `bool Driver::ejecucionFinalizada() const`
  Devuelve `true` si la calculadora finalizó la ejecución del programa.
- `void Driver::ejecutarInstruccionActual()`
  Ejecuta una (única) instrucción.
- `int Driver::topePila() const`
  Devuelve el tope de la pila. Si la pila está vacía, devuelve 0.
- `int Driver::valorVariable(string x) const`
  Devuelve el valor actual de la variable x en la calculadora.
- `int Driver::valorHistoricoVariable(string x, int t) const`
  Devuelve el valor histórico de la variable x en el instante t.
- `int Driver::instanteActual() const`
  Devuelve el valor del instante actual.

2. Pautas de entrega

Se debe enviar un `zip` a la casilla `algo2.dc+tp3@gmail.com` con fecha límite el miércoles 11 de julio. Además de la entrega digital, todos los integrantes del grupo deberán participar de la entrega presencial, cualquiera de los siguientes miércoles: {27 de junio, 4 de julio, 11 de julio} en el horario habitual del laboratorio. El asunto del mail debe constar de los números de libreta de todos los integrantes del grupo, separados por punto y coma (`";"`). Por ejemplo `"432/01; 123/45; 12/34; 543/21"`.