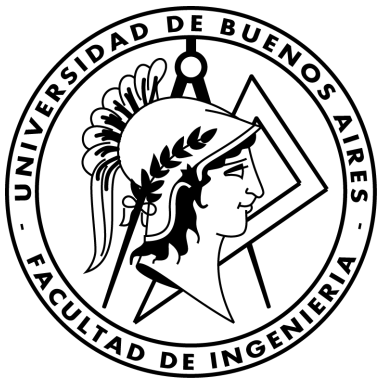

Universidad de Buenos Aires

Facultad de Ingeniería

2° Cuatrimestre de 2020

95.12

Algoritmos y Programación 2

Trabajo práctico

programación C++

Integrantes: *Padrón:*
Sanchez, Marcelo Fernando <marce_chez@msn.com> 87685
Acuña, Sofía Maribel <sacunia@fi.uba.ar> 98764

12 de noviembre de 2020

Índice

1. Enunciado

2. Desarrollo

2.1. Descripción de alto nivel

Para cumplir con los requerimientos de diseño se organizó el programa en 4 módulos con funcionalidades bien definidas:

- Módulo 1:*parser* para entrada de argumentos por consola.
- Módulo 2: lectura, verificación y carga de datos en el bloque.
- Módulo 3: cálculo y carga del *Header* de bloque.
- Módulo 4: escritura en archivo o salida estándar, según corresponda.

2.2. Clases

Se crearon 2 clases para manipular los datos en forma ordenada, y poder escalar fácilmente distintos tamaños de bloque posibles. Estos son: Clase Block y Clase Transaction. Se describen con más detalle a continuación.

2.2.1. Clase Block

Esta clase tiene como finalidad agrupar todos los campos necesarios del bloque generado, se compone de 5 atributos:

- string prev_block. Es el *Hash* del bloque anterior, a efectos de cumplir con el propósito del enunciado se le asignó el valor "fff....ff"sin acceso para modificación.
- string txns_hash. Es el *Hash* de todas las transacciones del bloque, se calcula en el módulo 3.
- size_t bits. Dificultad para calcular el *Hash* del ítem anterior.
- size_t nonce. Campo para modificar libremente, en este trabajo se modifica incrementando de a 1 valor en el Módulo 3.
- size_t txn_count. Indica la cantidad total de transacciones, en el programa se modifica simultáneamente con la carga de datos. En el Módulo 2.
- Array <Transaction>txns. Transacciones del bloque. Se implementó como un arreglo de clases Transaction para mayor claridad y modularidad de los datos.

Los métodos en esta clase son todos públicos y se componen de 3 variantes para el constructor, 1 *geter*, 1 *seter* y setNonce. Encargado de obtener los atributos nonce y txns_hash una vez cargados todos los datos y al bloque.

2.2.2. Clase Transaction

Esta clase se diseñó para poder trabajar directamente con los datos (*stream*) pasados al programa. Se compone de los atributos:

- size_t n_tx_in. Número de transacciones de entrada (inputs).
- Array <input_t>inputs. Arreglo de estructuras tipo input_t(*).
- size_t n_tx_out. Número de transacciones de salida (outpus).
- Array <output_t>outputs. Arreglo de estructuras tipo output_t(**).

(*) se definió una estructura tipo input_t, se muestra en el listado 1.

Listado 1 – Definición de las estructuras input_t y outpoint_t

```
1 typedef struct{
2         std::string tx_id;// el hash de la transaccion de donde este
              input toma fondos, 64 bytes fijos.
3         size_t idx;//idx sirve de indice sobre la secuencia de outputs
              de la transaccion con hash tx_id
4         std::string addr;//direccion de origen de los fondos, 64 bytes
              fijos(es un hash)
5 } input_t;
```

(**) Similar a la estructura input_t se definió una estructura tipo output_t, se muestra en el listado 2.

Listado 2 – Definición de las estructuras output_t

```
1 typedef struct{
2         float value = 0;
3         std::string addr; //direcci n destino de los fondos, 64 bytes
              fijos (es un hash)
4 } output_t;
```

2.3. Compilación

Para compilar utilizamos:

```
gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)
```

3. Conclusiones

4. Apéndice