**Universidad de Puerto Rico, Recinto de Río Piedras**
**Departamento de Ciencia de Cómputos**

# CCOM 3033 – Introducción a la Programación de Computadoras

Profa. Marie Lluberes
Otoño 2020

**PROYECTO FINAL – ÚLTIMA ETAPA**
Fecha: 11/29/2020
**Entrega: 12/11/2020**
**LA HORA DE ENTREGA APARECERÁ EN MOODLE.**

A continuación, las especificaciones finales de todo lo que debe contener y hacer su proyecto. El mismo ya debe poder hacer la mayor parte de lo que aquí se especifica fruto de los trabajos parciales anteriores.

1. Debe LEER el archivo de texto que se aneja, el cual contiene la información bancaria necesaria para los cómputos. No debe crear el archivo de texto; el mismo se le está proporcionando para que verifique que puede leerlo.

2. El nombre del archivo NO se le preguntará al usuario; TAMPOCO estará escrito directamente en el código. El nombre del archivo será un argumento durante "runtime". Para ello, la función main() debe contener los parámetros en el header:
   `int main(int argc,const char *argv[])`
   - Estos parámetros son la cantidad de argumentos a recibir por main() y un arreglo con dichos argumentos. El nombre del archivo estará en la segunda posición del arreglo argv[], así que cuando deseé referirse al nombre del archivo debe usar argv[1]. Úselo igual que si argv[1] fuera una variable de tipo string. NO use argv[0] ya que esta posición la ocupa el nombre de su programa.
   - Compile su programa como de costumbre. Cuando lo ejecute, en el "command line" debe añadir el path del archivo que va a leer luego del nombre del ejecutable. Ej: si su ejecutable se llama "ath" y el archivo a leer es "/Users/Miguel/docBanco.txt", entonces ejecuta como:
     `./ath /Users/Miguel/docBanco.txt`
     Note que hay un espacio entre el nombre del ejecutable y el path del archivo.

3. Cada vez que se ejecute, el programa debe desplegar (no escribir en archivo) un mensaje de bienvenida identificando el banco con el nombre que aparecen en el archivo (no lo escriba directamente en el código ni se lo pregunte al usuario).

4. A menos que se especifique lo contrario, toda la información que se produzca debe ANEJARSE al archivo de texto provisto. Es decir, se manejará un solo archivo para lectura y escritura, no dos archivos distintos.

5. El programa debe escribir en file un número de ATH cada vez que se ejecute. Transacciones realizadas durante la misma ejecución representan el mismo acceso del usuario a la ATH, así que deben tener el mismo número.
    a. Utilice valores aleatorios (random) para asignar un número distinto a la ATH cada vez que la use. Los números de las ATHs deben ser entre 100 y 500.

6. El programa debe pedirle al usuario su "PIN number". Debe entonces validar que el pin es igual al que existe en el archivo.
    a. De ser incorrecto, debe pedirlo de nuevo, dando tres oportunidades en total antes de "bloquear" el uso de la ATH. Dígale al usuario cuántas oportunidades le quedan cada vez.
    b. De ser correcto, le da la bienvenida al usuario por su nombre, el cuál debe ser leído del archivo y no escrito directamente en el código.

7. El programa debe desplegar un menú con las opciones que puede realizar en esa ATH.
    a. Las mismas son:
        i. Deposito
        ii. Retiro
        iii. Verificar balance
        iv. Realizar búsqueda
        v. Terminar.
    b. Debe asegurarse que la opción entrada es válida.
    c. De ser depósito debe preguntar si es cheque o efectivo.
    d. Debe asegurarse que el depósito es mayor que 0. Si es cheque, puede aceptar cualquier cantidad hasta $5,000.00 (incluyendo decimales). Si es en efectivo, puede aceptar solo billetes (de cualquier denominación) y cualquier cantidad.
    e. Debe asegurarse que el retiro es mayor que 0, que solo retira una cantidad en billetes de $10 y que tiene balance suficiente para retirar.

8. Luego de completar cada transacción debe mostrarle al usuario su balance actualizado.

9. Luego de completar cada transacción debe preguntarle al usuario si desea ejecutar alguna otra transacción y repetir el proceso, si elige que sí.

10. Debe mostrar mensaje de despedida del usuario cuando termina la ejecución.

11. Las transacciones que se anejan en el archivo deben contener lo siguiente:
    a. Descripción de las transacciones realizadas: número de ATH, nombre de la transacción (Deposito, Retiro, Verificación de balance), cantidad de la transacción, balance luego de realizar la misma y fecha y hora de la transacción. Puede usar cualquier orden para las columnas.
    b. Debe añadir lo descrito anteriormente CADA VEZ que use el programa. Es decir, debe anejar las transacciones al mismo archivo luego de cada ingreso.

c. EL BALANCE DEBE SER ACTUALIZADO DE LAS TRANSACCIONES ANTERIORES. Es decir, usará el balance inicial que dice el archivo provisto para LA PRIMERA VEZ que se ejecute el programa. Luego de realizar una transacción, el balance será actualizado. Las transacciones subsiguientes tomarán el balance de la transacción anterior.
d. Para desplegar la fecha y hora, use lo explicado en asignaciones anteriores.

12. Debe implementar un algoritmo de búsqueda:
    a. Usando como criterio el tipo de transacción (depósito o retiro). Esta opción estará en el menú.
    b. El criterio lo proporcionará el usuario; debe preguntarle.
    c. Cada búsqueda debe DESPLEGAR SOLAMENTE todas las transacciones que corresponden a dicho criterio, con toda la información que corresponde a cada transacción.
    d. Si no hay ninguna transacción que corresponda al criterio, debe desplegar un mensaje indicándolo (por ejemplo: “No se encontraron transacciones de [transacción]”).

13. ESPECIFICACIONES DEL CÓDIGO:
    a. Debe estar modularizado: debe usar no menos de 10 funciones. Procure que una función no tenga mas de 5 parámetros.
    b. Las funciones deben usar tanto “pass by value” como “pass by reference”. NO DEBE USAR VARIABLES GLOBALES.
    c. Debe usar structs para organizar la data que lee y escribe (cada fila es un struct).
    d. Toda la data debe estar ordenada en un arreglo de structs. Puede usar vectores de structs.
    e. Debe separar la implemetación de la especificación (declaración). Para ello, creará dos archivos adicionales según se explica a continuación.
    f. Cree un archivo [nombre que desee].h con las declaraciones de los structs y los prototipos de las funciones usadas. **Cada función debe estar documentada con su propósito, descripción de sus parámetros (glosario de variables) y de lo que devuelve.**
    g. Cree un archivo [nombre que desee].cpp con la implementación de las funciones. Es decir, aquí deben estar las definiciones de las funciones. Usualmente, este archivo y el .h tienen el mismo nombre. **Documente brevemente cada función.**
    h. El archivo con la implementación ([nombre que desee].cpp) y el archivo que contiene a main() deben contener el archivo .h como header file, usando la extensión .h y entre “ “. Es decir, debe añadir:
    `#include "[nombre que desee].h"`

    i. Lo anterior puede hacerlo haciendo “cut and copy” del archivo actual que hizo conteniendo las funciones.

j. Para compilar el proyecto, debe incluir todos los archivos .cpp que contenga el mismo (en nuestro caso, solo dos). En command line, use (la extensión ".exe" sólo se usa en los sistemas Windows):
`g++ -o project.exe [main file].cpp [nombre que desee].cpp`

A continuación, ejemplos de archivo de entrada, despliegue en pantalla y archivo de salida:

1. Ejemplo de archivo de entrada:

```
Banco Impopular Del Pueblo

Account_No.      1234567
Name_Last:       Lisa Gomez
Soc_Security:    123-45-6789
PIN_number:      1234
Init_Balance:    1000.00
```

Este archivo identifica al usuario como Lisa Gómez, con PIN 1234 y un balance inicial de $1000.00. A partir de ese balance, se actualizarán las transacciones que realice en cada ingreso.

2. Ejemplo del archivo de salida:

```
Banco Impopular Del Pueblo

Account_No.      1234567
Name_Last:       Lisa Gomez
Soc_Security:    123-45-6789
PIN_number:      1234
Init_Balance:    1000.00
Act_Balance:     1127.91

ATH      TRANSACCION           CANTIDAD        NUEVO BALANCE              FECHA
427      Deposito                67.91              1067.91       Sun Oct 18 19:40:14 2020
427      Deposito                67.00              1134.91       Sun Oct 18 19:40:28 2020
427      Deposito                43.00              1177.91       Sun Oct 18 19:40:58 2020
427      Retiro                  60.00              1117.91       Sun Oct 18 19:41:10 2020
427      Balance Check         1117.91              1117.91       Sun Oct 18 19:41:15 2020
146      Deposito                90.00              1207.91       Tue Oct 20 13:20:18 2020
146      Balance Check         1207.91              1207.91       Tue Oct 20 13:21:02 2020
392      Retiro                  80.00              1127.91       Fri Oct 23 16:11:10 2020
392      Balance Check         1127.91              1127.91       Fri Oct 23 16:11:46 2020
```

3. Una búsqueda de transacciones de retiro desplegaría en pantalla:

```
ATH      TRANSACCION           CANTIDAD        NUEVO BALANCE              FECHA
427      Retiro                  60.00              1117.91       Sun Oct 18 19:41:10 2020
392      Retiro                  80.00              1127.91       Fri Oct 23 16:11:10 2020
```