UNIVERSIDAD PRIVADA BOLIVIANA

-

MATERIA: ALGORITMICA 1

PROYECTO: CAJA REGISTRADORA

ARIANE GARRETT BECERRA

ALEJANDRA ANDREA GARCIA BERNAL

CAMILA XIMENA GUZMAN SALVATIERRA

INGENIERÍA DE SISTEMAS COMPUTACIONALES

LA PAZ – BOLIVIA

2021

ÍNDICE

INTRODUCCIÓN

INTRODUCCIÓN

Para el proyecto final de la materia de algorítmica 1, decidimos trabajar en un algoritmo que funciona como la caja registradora de una tienda o supermercado. Con este algoritmo, podemos definir el cambio que debería dar el cajero a un cliente en base al monto que debe pagar y el monto con el que paga. Además, el algoritmo nos ayuda a identificar la mínima cantidad de monedas o billetes utilizados al momento de dar cambio y nos da todas las formas posibles en las que podemos dar el cambio según el billete o moneda disponible en la tienda.

1. OBJETIVO DEL PROYECTO

Implementar un algoritmo visto en clase para automatizar una situación de la vida diaria, con el fin de solucionar un problema de tiempo, en nuestro caso, dar el cambio adecuado en una tienda o supermercado.

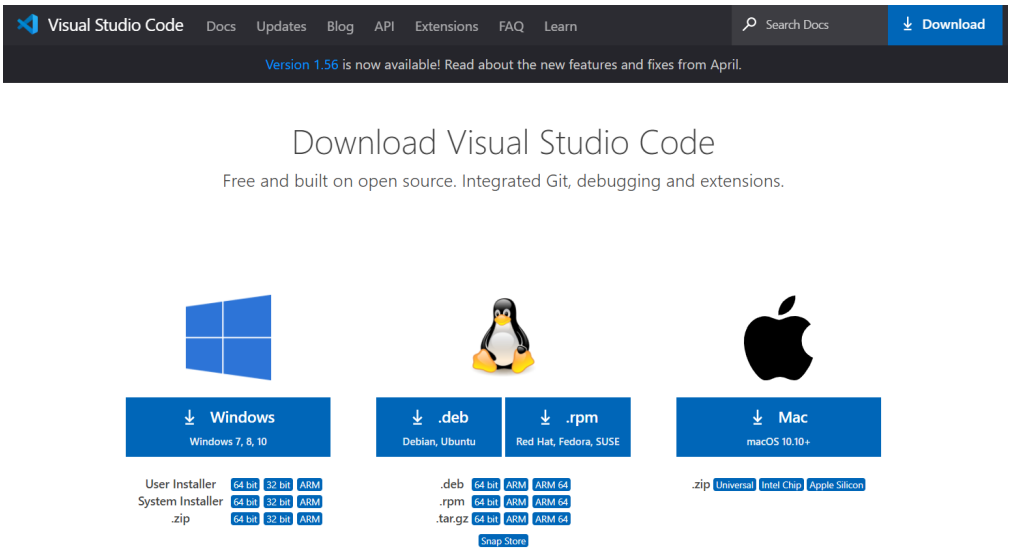
2. PROCESO DE INSTALACIÓN

Este proyecto fue desarrollado en el lenguaje de programación C++ y fue compilado en su totalidad en el software Visual Studio Code; a continuación los pasos a seguir para una instalación exitosa de este software. (Para Windows)

ICONO DEL SOFTWARE

Paso #1: Ve a la página de Microsoft Visual Studio Code y haz clic en el botón “Descargar”, para descargar el instalador. Selecciona el botón dependiendo de las características de tu ordenador. En el caso de tener Windows, deberá seleccionar la opción de “System Installer”.

- Link de descarga: https://code.visualstudio.com/download



Paso #2: Abre el archivo de instalación .exe en tu carpeta de descargas para iniciar la instalación.

Paso #3: Sigue las instrucciones de instalación para poder tener instalado correctamente Microsoft Visual Studio Code.

Paso #4: Descargar el compilador C/C++, en este caso es MinGW.

- Link de descarga: https://sourceforge.net/projects/mingw/files/

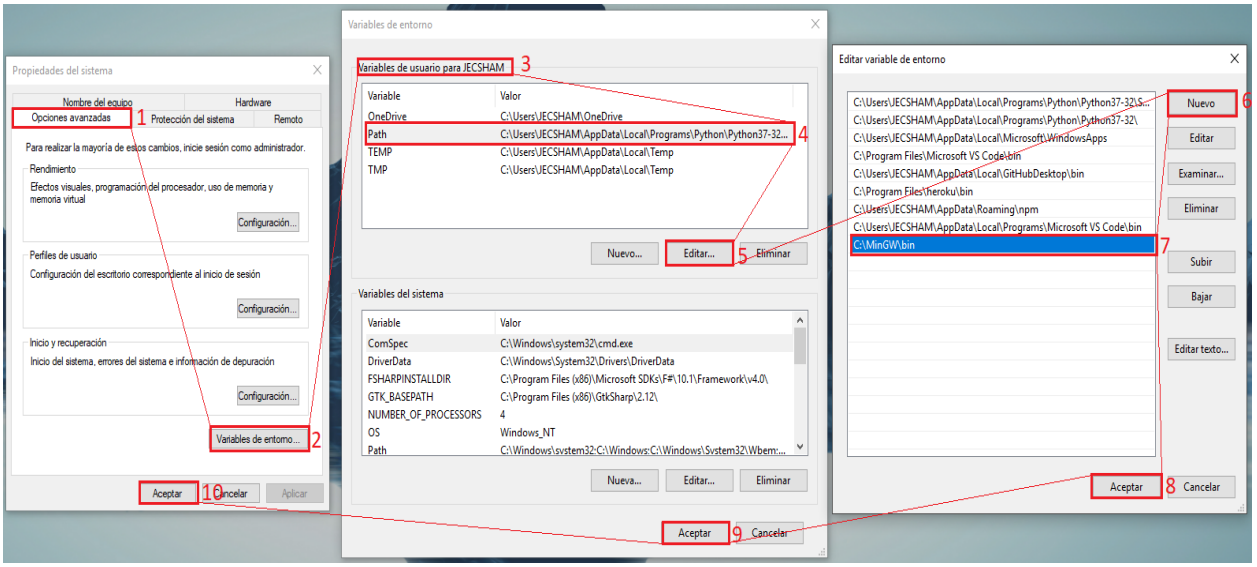
Paso #5: Instalar extensión para Visual Studio Code: “C/C++” (Formato para C/C++).

Paso #6: Instalar extensión para Visual Studio Code: “Code Runner” (Compilador con un botón).

Paso #8: Seleccionamos todas las pestañas de “Basic Setup” y marcamos todos.

Paso #9: Pulsamos en “Installation” y “Apply Changes”.

Paso #10: seguimos el esquema que a continuación les dejo, en el paso 7 deben colocar esta ruta “C:\MinGW\bin” solo si realizaron la instalación por defecto de otro modo sería “<su ruta>\bin”



Paso #11: Con las extensiones instaladas del ***Paso #5*** y ***Paso #6*** podemos compilar y hacer correr nuestro programa, como nuestro proyecto tiene un “INPUT” para ingresar los requerimientos, en este caso ingresamos manualmente el input en la parte de “terminal”.

Paso #12: Existen 2 consolas donde puedes ver el resultado o lo que sale del programa pero solo en una puedes ingresar y input incluido y es en la “terminal”, para habilitar esta función vamos al “Code Runner” y marcamos donde dice “ejecutar en terminal”

Paso #13: Habilitar input y output en la terminal debe ir a la parte de "vscode" que se encuentra al lado izquierdo donde están todas sus tareas, presiona “settings.json” y copiar el siguiente comando: `"code-runner.runInTerminal": true.`

Si siguió los pasos correctamente, debería ser posible ejecutar el proyecto sin ningún problema

3. EXPLICACIÓN DEL ALGORITMO

El algoritmo implementado fue el de CoinChange. Este algoritmo se encarga de calcular la cantidad mínima de monedas y/o billetes utilizados para dar una cierta cantidad de dinero, o para determinar las maneras en las que se puede dar una cantidad de dinero dependiendo de las monedas y/o billetes que se tengan.

Aquí podemos ver el código para hallar la mínima cantidad de monedas y/o billetes utilizados para dar cierto monto de dinero.

```
long long dpMin[1000000];
long long coinChangeMin(long long value){
    if(value == 0){
        return 0;
    }
    if(dpMin[value] = -1){
        long long minValue = 1e9;
        for (int i = 0; i < length(coins); i++){
            if(value - coins[i] >= 0){
                minValue = min(minValue, 1 + coinChangeMin(value - coins[i]));
            }
        }
        dpMin[value] = minValue;
    }
    return dpMin[value];
}
```

A partir de este código y de las monedas que tengamos (que en nuestro caso son solamente billetes de 200Bs, 100Bs, 50Bs, 20Bs, 10Bs y monedas de 5Bs, 2Bs, y 1Bs), podemos determinar la mínima cantidad de monedas y/o billetes utilizados para dar cierto monto de dinero.

Por ejemplo, si el cliente debe pagar 290Bs y entrega 300Bs en efectivo, el cambio que debe recibir será de 10Bs, y la mínima cantidad de billetes usados para darle el cambio es de 1, ya que se puede utilizar solamente un billete de 10Bs.

El algoritmo de CoinChange también nos ayuda a determinar las maneras en las que se puede dar cierta cantidad de dinero, el código lo podemos ver a continuación:

```
int dp[9][1000];
int coinChange(int value){
    memset(dp, 0, sizeof dp);
    for (int i = 1; i <= value; i++){
        dp[0][i] = 0;
    }
    for (int i = 0; i < length(coins); i++){
        dp[i][0] = 1;
    }
    for (int i = 1; i <= length(coins); i++){
        for (int j = 0; j <= value; j++){
            if(j - coins[i-1] >= 0){
                dp[i][j] = dp[i][j-coins[i-1]] + dp[i-1][j];
            } else {
                dp[i][j] = dp[i-1][j];
            }
        }
    }
    return dp[length(coins)][value];
}
```

Además, después de haber calculado las diferentes maneras de dar el cambio, podemos imprimir estas para poder decidir cuál de estas nos conviene utilizar. El código para realizar esto es el siguiente:

```
vector<int> manera;
void printChanges(int i, int j){
    if(dp[i][j] == 0){
        return;
    }
    if(j == 0){
        for (int i = 0; i < manera.size(); i++){
            cout << "[" << manera[i] << "]";
        }
        cout << endl;
        return;
    }
    if(dp[i][j] > dp[i-1][j]){
        manera.push_back(coins[i-1]);
        printChanges(i, j-coins[i-1]);
        manera.pop_back();
    }
    printChanges(i-1, j);
}
```

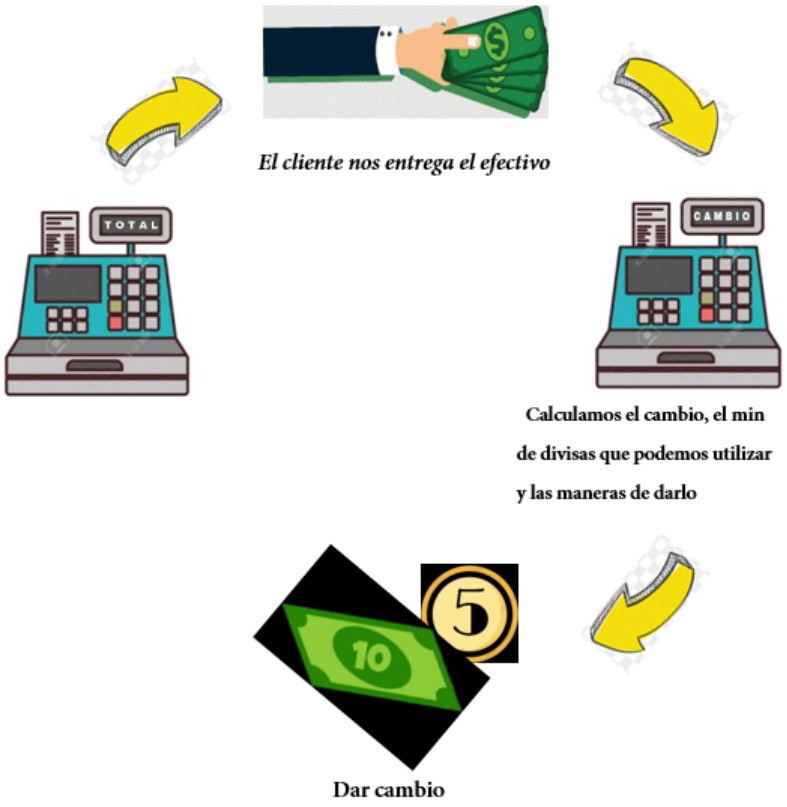
Entonces, retomando el anterior ejemplo, al tener que dar 10Bs de cambio, podemos calcular el número de maneras con las que se puede dar esta cantidad de dinero, que en este caso serían 11 maneras de dar el cambio.

Y con el método printChanges(), podemos imprimir estas 11 posibilidades para decidir cuál utilizar. En este caso, las 11 posibilidades serían:

1. [10]
2. [5][5]
3. [5][2][2][1]
4. [5][2][1][1][1]
5. [5][1][1][1][1][1]
6. [2][2][2][2][2]
7. [2][2][2][2][1][1]
8. [2][2][2][1][1][1][1]
9. [2][2][1][1][1][1][1][1]
10. [2][1][1][1][1][1][1][1][1]
11. [1][1][1][1][1][1][1][1][1][1]

Donde cada [] representa el monto en billetes o monedas utilizados para dar el cambio.

Diagrama de funcionamiento del algoritmo:



4. CONCLUSIONES

Los algoritmos que vimos en clase no son solo cosas abstractas y pueden ser aplicados en un problema de la vida diaria brindando una solución eficiente para nuestro tiempo. Aplicando el algoritmo de CoinChange en nuestro proyecto logramos hacer el proceso de restar y sumar grandes cantidades de dinero en solo segundos, sin cometer el error de dar cambio por demás o dar menos, porque aparte de calcular el cambio exacto, nos dice con qué monedas/billetes disponibles podemos dar es el cambio.