Tecnológico de Monterrey

Desarrollo de aplicaciones avanzadas de ciencias computacionales

Actividad Final Mini-Proyecto

Luis Omar Leyva Navarrete A01570367

Campus Monterrey

Máquina Virtual

La máquina virtual se desarrolló en un archivo independiente de la gramática. No se utilizó alguna librería específica para su desarrollo. La máquina virtual se compone por los siguientes pedazos de código.

1. Mapa de Memoria

El mapa de memoria utiliza un diccionario que utiliza un proceso de hashing para simular una dirección de memoria única en la máquina virtual.

```python
MAPA_MEMORIA = {}
```

También se declaró una función en la cual se regresa el mapa de memoria al archivo principal para ser impreso en un archivo de texto.

```python
def get_mapa_memoria():
    return MAPA MEMORIA
```

2. Revisión de variables

La función de check_variable recibe una variable para revisar si ya fue guardada en la memoria. En caso de que no exista se agrega a la memoria con la excepción de un id debido a que este método se utiliza para el resto de los operadores con la excepción del operador de asignación.

```python
def check_variable(var):
    try:
        if not MAPA_MEMORIA.keys().__contains__(hash(var)):
            if var == tk.id_:
                raise Exception(
                    "La variable "
                    + var
                    + " no ha sido declarada o no contiene un valor asignado"
```

```
                )
            else:
                MAPA_MEMORIA[hash(var)] = var
    except Exception as e:
        print(e)
        exit()
```

3. **Borrado de memoria temporal**

Se creó una función que elimina las variables temporales de la memoria para ser liberadas y reutilizadas en próximos procesos.

```python
def delete_temp_memoria(var1, var2):
    if var1 == "temp":
        del MAPA_MEMORIA[hash(var1)]
    if var2 == "temp":
        del MAPA_MEMORIA[hash(var2)]
```

4. **Ejecución de cuádruplos**

En esta función es donde se ejecutan las operaciones y asignaciones de memoria necesarias para la funcionalidad del lenguaje. Se utilizó un Match Case del lenguaje de python, el cual es un equivalente para el Switch de otros lenguajes de programación. Los resultados son grabados en un archivo de texto.

Se utiliza un loop de while que leerá la fila de cuadruplos línea por línea. Un contador es utilizado para saber cuál línea de cuádruplo está siendo ejecutada. El código revisa el primer elemento de la lista, el cual es un operador, y utiliza el Match Case para elegir qué instrucción seguir. Una vez elegida la función se leen los dos operandos y se revisa su existencia en memoria, en caso de que no exista se levanta un error o se agregan a memoria. Después se ejecuta la acción definida por el operador y el resultado es guardado utilizando el operando del cuarto elemento de la lista. Para terminar se eliminan las variables temporales de la memoria y se aumenta el contador.

Python

```python
def execute_quad(quadruples):
    open("resultado_ejecucion/resultados.txt", "w")
    length = len(quadruples)
    i = 0
    while i < length:
        line = quadruples[i]
        match line[0]:
            case "+":
                check_variable(line[1])
                check_variable(line[2])

                MAPA_MEMORIA[hash(line[3])] = (
                    MAPA_MEMORIA[hash(line[1])] + MAPA_MEMORIA[hash(line[2])]
                )
                delete_temp_memoria(line[1], line[2])

            case "-":
                check_variable(line[1])
                check_variable(line[2])

                MAPA_MEMORIA[hash(line[3])] = (
                    MAPA_MEMORIA[hash(line[1])] - MAPA_MEMORIA[hash(line[2])]
                )

                delete_temp_memoria(line[1], line[2])
            case "*":
                check_variable(line[1])
                check_variable(line[2])

                MAPA_MEMORIA[hash(line[3])] = (
                    MAPA_MEMORIA[hash(line[1])] * MAPA_MEMORIA[hash(line[2])]
                )

                delete_temp_memoria(line[1], line[2])
```

```
        case "/":
            check_variable(line[1])
            check_variable(line[2])

            MAPA_MEMORIA[hash(line[3])] = (
                MAPA_MEMORIA[hash(line[1])] / MAPA_MEMORIA[hash(line[2])]
            )

            delete_temp_memoria(line[1], line[2])

        case "=":
            if not MAPA_MEMORIA.keys().__contains__(hash(line[1])):
                MAPA_MEMORIA[hash(line[1])] = line[1]

            MAPA_MEMORIA[hash(line[3])] = MAPA_MEMORIA[hash(line[1])]
            if type(line[1]) == str:
                if line[1][0:4] == "temp":
                    del MAPA_MEMORIA[hash(line[1])]

        case "cout":
            check_variable(line[3])

            # Escribimos los resultados en un archivo
            with open("resultado_ejecucion/resultados.txt", "a") as file:
                file.write(str(MAPA_MEMORIA[hash(line[3])]) + "\n")

            if type(line[3]) == str:
                if line[3][0:4] == "temp":
                    del MAPA_MEMORIA[hash(line[3])]

        case "concat":
            check_variable(line[1])
            check_variable(line[2])
```

```
            left = MAPA_MEMORIA[hash(line[1])]
            if type(left) != str:
                left = str(MAPA_MEMORIA[hash(line[1])])
            else:
                left = left.lstrip('"').rstrip('"')

            right = MAPA_MEMORIA[hash(line[2])]
            if type(right) != str:
                right = str(MAPA_MEMORIA[hash(line[2])])
            else:
                right = right.lstrip('"').rstrip('"')

            MAPA_MEMORIA[hash(line[3])] = left + " " + right

            delete_temp_memoria(line[1], line[2])

        case "!=":
            check_variable(line[1])
            check_variable(line[2])

            MAPA_MEMORIA[hash(line[3])] = (
                MAPA_MEMORIA[hash(line[1])] != MAPA_MEMORIA[hash(line[2])]
            )

            delete_temp_memoria(line[1], line[2])

        case ">":
            check_variable(line[1])
            check_variable(line[2])

            MAPA_MEMORIA[hash(line[3])] = (
                MAPA_MEMORIA[hash(line[1])] > MAPA_MEMORIA[hash(line[2])]
            )
```

```
            delete_temp_memoria(line[1], line[2])

        case "<":
            check_variable(line[1])
            check_variable(line[2])

            MAPA_MEMORIA[hash(line[3])] = (
                MAPA_MEMORIA[hash(line[1])] < MAPA_MEMORIA[hash(line[2])]
            )

            delete_temp_memoria(line[1], line[2])

        case "GOTO":
            i = line[3] - 2

            if type(line[3]) == str:
                if line[3][0:4] == "temp":
                    del MAPA_MEMORIA[hash(line[3])]
                    del [line[3]]

        case "GOTOF":
            if not MAPA_MEMORIA[hash(line[1])]:
                i = line[3] - 2

                if type(line[3]) == str:
                    if line[3][0:4] == "temp":
                        del MAPA_MEMORIA[hash(line[3])]
                        del [line[3]]

        case "GOTOV":
            if MAPA_MEMORIA[hash(line[1])]:
                i = line[3] - 2
                if type(line[3]) == str:
```

```
                    if line[3][0:4] == "temp":
                        del MAPA_MEMORIA[hash(line[3])]
                        del [line[3]]
        i += 1
```