TEC | Tecnológico de Costa Rica

Área académica de Ingeniería en Computadores

Algoritmos y Estructuras de Datos 2

Planificación Proyecto 1

Profesor:
Luis Antonio González Torres

Estudiantes:
Paola Quirós Chinchilla
Luis David Richmond Soto

Primer Semestre, 2021

Tabla de contenido

Descripción del problema

El proyecto consiste en el diseño e implementación de un IDE que permite interpretar y ejecutar el pseudo lenguaje C! con manejo de memoria, mediante las diferentes secciones que este posee se puede editar texto, imprimir los datos especificados por medio de la llamada a cout, además permite tener un control detallado de los errores internos producidos durante la ejecución así tambien se puede observar el estado de la RAM en cuanto a la dirección de memoria, valor, etiqueta y conteo de referencias que posee, conforme cada línea de C! se ejecuta.

Algoritmos desarrollados

Split

Por medio de esta función se realiza la separación del código en líneas individuales para así lograr generar un vector de string con el cual se realizara el parse de los datos ingresado.

```
void StringParse::SplitString(const string& texto, char del, vector<string>& v){

   string::size_type i = 0;
   string::size_type j = texto.find(del);

   while (j != string::npos){
      v.push_back(texto.substr(i,j-i));
      i = ++j;
      j = texto.find(del,j);

      if (j == string::npos){
         v.push_back(texto.substr(i,texto.length()));
      }
   }

   if (v.size() == 0){
      v.push_back(texto.substr(0,1));
   }
}
```

Add

Función encargada de agregar los datos ingresados por el usuario a la memoria como un nodo el cual tiene asignando el valor y nombre dado al dato, asi mismo hace el control del Garbage Collector para verificar cuando esta vacio y cuando una se debe remover un dato contenido en el vector.

```
void AddData(T data, string name) {
   if(garbage.empty()){
      Node<T> *temp = new(mem+size) Node<T>(data,name);
      size++;
   } else{
      Node<T> *temp = new(mem+garbage.back()) Node<T>(data,name);
      garbage.pop_back();
   }
}
```

Operaciones

Algortimo que realiza la resolución de operaciones basado en la ubicación de carecteres especiales (+,-,*,/) en cada linea de código para poder resolver estas operaciones entre dos valores ingresados asi como cuando se indica una variable anteriormente ingresada por el usuario.

```
if(line.at(i) == "+"){

    valor1 = ObtenerNumero(line.at(i+1));

    if (valor1 != "error"){
        valor += stoi(valor1);
    } else{
        return "error";
    }
```

Manejo de Json

El uso de Json permite el manejo de los datos ingresados por el usuario para enviar estos hacia el servidor, y mediante el cual se la utilización para funciones como la muestra de los valores por medio de la interfaz con el uso de la RAM Live View, siendo esta una forma organizada del manejo de la información a intercambiar.

Estructuras de datos desarrolladas

Facade

El patrón Facade tiene la característica de ocultar la complejidad de interactuar con un conjunto de subsistemas proporcionando una interface de alto nivel, la cual se encarga de realizar la comunicación con todos los subsistemas necesarios, lo cual permitió realizar la conexión entre el servidor y el cliente para realizar las funciones asignadas para el manejo de los datos recibidos y enviados entre ambos.

Vector

El tipo de variable vector es una secuencia de objetos del mismo tipo almacenados consecutivamente en memoria que permite almacenar varios valores bajo una sola variable y realizar operaciones complejas con esta, esta estructura se utlizó para el manejo de los datos de dirección de memoria, valor, etiqueta y conteo de las referencias que se obtienen de las variables definidas por el usuario en el editor de C!.

Condicionales

Una estructura condicional permite decidir por cuál alternativa seguirá el flujo del programa dependiendo del resultado de la evaluación de una condición, este tipo de estructura se utilizó para verificar el tipo de dato ingresado para lograr clasificar los datos de acuerdo a este.

Diagrama de clases