Análisis Semántico

Agrega los siguientes atributos a la clase Nodo:

```
char tipoDato;
static TablaSimbolos *tablaSimbolos;
static string ambito;
```

Agrega el siguiente método a la clase Nodo:

```
virtual void validaTipos() {
      tipoDato= 'v';
      if (sig != NULL) sig->validaTipos();
};
```

Agrega el siguiente atributo a la clase Semantico (definida en semántico.h)

```
TablaSimbolos *tablaSimbolos;
```

Agrega los siguientes métodos a la clase semántico:

```
Semantico(){
      Nodo::tablaSimbolos= tablaSimbolos= new
      TablaSimbolos(&listaErrores);
}


void analiza(Nodo *arbol){
      this->arbol= arbol;
      arbol->validaTipos();

      tablaSimbolos->muestra();
      muestraErrores();
}
```

Agrega el siguiente método a la clase Tipo

```
char dimeTipo() {
      if ( simbolo.compare("int") == 0) return 'i';
      if ( simbolo.compare("float") == 0) return 'f';
      if ( simbolo.compare("string") == 0) return 's';
      if ( simbolo.compare("void") == 0) return 'v';
}
```

En el archivo principal.cpp agrega las siguientes líneas como variables globales:

```
TablaSimbolos * Nodo::tablaSimbolos= NULL;
string Nodo::ambito= "";
```

En el main del archivo principal.cpp agrega lo siguiente:

```
Semantico semantico;

semantico.analiza(arbol);
```

Antes de comenzar a definir las funciones de validaciones veamos los métodos con lo que ya cuenta la tabla de símbolos:

```
void agrega( ElementoTabla *elemento);
Este método inserta el elemento que se pasa como argumento en la sublista
que le corresponde utilizando la función de dispersión.

void muestra();

bool varGlobalDefinida(string simbolo);
bool funcionDefinida(string simbolo);
bool varLocalDefinida(string variable, string funcion);
void buscaIdentificador(string simbolo);
Esta función asigna a los atributos varLocal, varGlobal, funcion el
objeto que coincide con simbolo, si no encuentra nada pone los tres
valores en NULL.

void buscaFuncion(string simbolo);
```

Estos tres métodos se necesitan completar:

```
void agrega( DefVar *defVar);
void agrega( DefFunc *defFunc);
void agrega( Parametro *parametros);
```

Para hacer la validación semántica de las definiciones de variable agrega el siguiente método en la clase DefVar

```
void validaTipos(){
      tipoDato= tipo->dimeTipo();
      tablaSimbolos->agrega(this);

}
```

Agrega el siguiente método a la clase Tabla de Símbolos

```
void agrega( DefVar *defVar){
      char tipo= defVar->tipo->dimeTipo();
      Identificador *p= defVar->listaVar;
      ElementoTabla *elem;

      elem= new Variable(tipo, p->simbolo, Nodo::ambito);
      agrega(elem);
```

}

El método anterior no toma en cuenta si se están definiendo más de una variable, ni tampoco si ya estaban definidas. Para agregar la lista de variables será necesario recorrer la lista simplemente ligada (usando el campo sig). Para revisar si una variable local esta definida usa el método `varLocalDefinida,` si es una variable global usa `varLocalDefinida.`

Para hacer las validaciones de tipos será necesario modificar los métodos de las clases que heredan de expresión:

Por ejem. La siguiente línea

```
if (izq->tipoDato == 'i' && izq->tipoDato == der->tipoDato) tipoDato=
'i';
```

Para agregar un error a la lista de errores, dentro de los métodos de validación usa lo siguiente:

```
Nodo::tablaSimbolos->listaErrores->push_back("Error: variable local \"" +
p->simbolo + "\" redefinida");
```

Práctica:

- Completa los tres métodos agrega de la tabla de símbolos
- Define los métodos de validación para las clases que heredan de las expresiones.