Compiler Simulation Manual

Introduction

This project is a simulation for the compiler phases in practical way which (scanning , parsing , semantic analysis , immediate code generation , code optimization and final code generation).

Note that is not an interpreting but it's similar as both showed that line by line is complied but the main differences are :

1- Interpreted languages are transformed in another language and then compiled to machine code on the fly which mean not object code is generated just machine code directly and every time you run the program it do the same phases .

2- It may be similar as it that project you just allow to write one line in each run or a block of code in { multiple line of code } but that is not the main common characteristic in both as in that project we receive tokens and pares it then check the pare tree to generate the annotated tree then immediate code generation (three address code matrix) then we use c# langue to run the three address code matrix and do the two final steps which code optimization and code generation (object file) .

   The just remember that main objective of that project is to represent the compiling phases in practical way not to construct a real compiler for a secretin programing language .

Basics

```
>> 1+1

>> 1-2

>> 25/5

>> 5*5


>> true

>> false

>> true && false

>> true || false

>> true && true

>> false && false
```

Variables

```
var VarName = value /* can't change the type of the var without using the wordkey var again
*/

var x = 10

x= 11

x = ‘’hello” /* that causes error*/

var x = “hello” /*correct*/
```

constants

let Varname = value /* can't change the value of the var without using the wordkey let again */

let x = 10

x = 22 /* that causes error*/

let x = 22 /*correct*/

if statements

```
if condition
        one line


if condition
{
        Many lines
}
```

```
if condition
{
        Many lines
} else
{
        Many lines
}
=================================================================
if condition
{
        Many lines
} else if condition
{
        Many lines
} else
{
        Many lines
}
```

Loops (for, while , do while)

```
while condition
lineOfCode
```

```
while condition
{
        Many lines of code
}
```

```
do{
        lines of code
}while condition
```

```
for i=value1 to value2
line of code
```

```
for i=value1 to value2
{
        line of code
}
```

/* in for we can use any variable but without var and should be assigned in for statement also can use continue and break in the loops */

Define functions

```
/* void function without arguments   */
function funcName ()
{
}
```

```
/* void function with arguments  */
function funcName (arg1:ArgType , arg2:ArgType)
{
}
```

```
/* return function with arguments   */
function funcName (arg1:ArgType , arg2:ArgType) : returnType
{

        return (var OR value)
}
```

```
/* return function without arguments   */
function funcName () : returnType
{

        return (var OR value)
}
```

built-in functions

```
print("string") /* to print string and must take string
 string( value ) /* convert the value to string
int(" string ") /* convert the value to integer
intput() /* to revise an input from the user */
```