Compiler Project Manual

Submitted by:

Saimoon Al Farshi Oman
Roll: 1807018

Course No: CSE 3212
Course Title: Compiler Design Laboratory

Under the supervision of:

Dola Das
Assistant Professor
Department of Computer Science and Engineering
Khulna University of Engineering & Technology

Dipannita Biswas
Lecturer
Department of Computer Science and Engineering
Khulna University of Engineering & Technology

Compiler Project Manual

Sl no.	Keyword/Symbol	Description
1	`#ImporT >file_name.h<`	For including library file Ex: #ImporT >MylibrarY.h<

2	`>IntegeR<`
	
	`>CharacteR<`
	
	`>LonG<`
	
	`>FloaT<`
	
	`>FloaT64<`
		Defining Integer
		Ex: >IntegerR< num1 := 10
		
		Defining Character
		Ex: >CharacteR< ch := A
		
		Defining Long
		Ex: >LonG< num1 := 9999
		
		Defining Float (32 bit)
		Ex: >FloaT< num1 := 7.77
		
		Defining Float (64 bit)
		Ex: >FloaT< num1 := 1117.77

3	`+`
	
	`-`
	
	`*`
	
	`/`
	
	`%`
	
	`**`
		Defining algebraic operation. +, -, *, / is used in the same algebraic way. % is used for modular operation. ** means power operation like 2**3 mean 2^3
		
		Ex:
		num3 := num1 + num2
		num3 := num1 - num2
		num3 := num1 * num2
		num3 := num1 / num2
		num3 := num1 % num2
		num3 := num1 ** num2

4	`>`
	
	`<`
	
	`>=`
	
	`<=`
	
	`EquaL`
	
	`&AnD`
	
	`|OR`
		Defining Logical operation
		
		`>` : Greater than
		Ex : num1 > 7
		
		`<` : Less than
		Ex : num1 > 7
		
		`>=` : Greater or equal
		Ex : num1 >= 7
		
		`<=` : Less or equal
		Ex : num1 <= 7

	!NoT	EquaL : Equal To Ex : num1 EquaL 7 &AnD : Logical AND operation Ex : num1 &AnD 7 |OR : Logical OR operation Ex : num1 |OR 7 !NoT : Logical NOT operation Ex : !NoT num1
5	:=	Assignment Operator: For assigning value in variable Ex: num3 := num1 + num2
6	{{ }} (( )) ;; ,, ::	{{ : Starting curly bracket }} : Ending curly bracket ((: Starting bracket)) : Ending bracket ;; : Defining end of statement ,, : Work same as comma separator :: : Work same as colon Ex: ->IF<- ((num1 EquaL 7)) {{ ->OuT<- ((7)) }} ->ElsE<- {{ ->OuT<- (("Not Found")) }}
7	->FoR<-	For defining for loop: Syntax: ->FoR<- ((initialization :: condition :: inc/dec)) {{ }}

		Ex: ->FoR<- ((num2 := 1 :: num2 < 5 :: num2 IncrementT1)) {{ }}
8	->WhilE<-	For defining while loop: Syntax: ->WhilE<- ((condition)) {{ }} Ex: num1 := 1 ->WhilE<- ((num1 <= 5)) {{ ->OuT<- (("Hello World")) num1 IncrementT1 }}
9	AnD OR XoR NoT	Defining bitwise operator: AnD : Bitwise and Ex: num3 := num1 AnD num2 OR : Bitwise or Ex: num3 := num1 OR num2 XoR : Bitwise xor Ex: num3 := num1 XoR num2 NoT : Bitwise not Ex: num3 := NoT num1
10	->SizeoF<-	Defining variable size in byte Syntax: ->SizeoF<-((Variable_name)) Ex: ->SizeoF<-((num3))
11	->->	For defining single line comment Ex:

		->-> This is a comment
12	>DataType<((size))[elements]	For defining sequence of same type of data which is array Syntax: array_name := >DataType<((array_size))[elements] Ex: arr := >IntegeR<((2))[1,, 2]
13	->IF<- ->EliF<- ->ElsE<-	Used for if else ladder Syntax: ->IF<- ((condition)) {{ }} ->EliF<- ((condition)) {{ }} ->ElsE<- {{ }}
14	->IN<- ((variable_name)) ->OuT<- (( ))	IN is used for user input Syntax: ->IN<- ((variable_name)) Ex : ->IN<- ((num2)) OuT is used for printing in console Ex : ->OuT<- (("Hello World\n"))
15	->SwitcH<- >CasE< value:: >DefaulT<::	>SwitcH< is used for switch case >Case< is used for comparing Ex: ->SwitcH<- ((num1)) {{ >CasE< 5:: ((->OuT<- ((5)))) >CasE< 7:: ((->OuT<- ((7)))) >DefaulT<:: ((->OuT<- (("Not Found"))))

		}}
16	->MaiN<- >BacK<	This indicates main function >Back< is used for returning from function Ex: >IntegeR< ->MaiN<-(()) {{ >BacK< 0 }}
17	>StrucT<	For creating a structure Ex: >StrucT< complex {{ >FloaT< real >FloaT< imag }}
18	Defining function Return_type func_name ((parameters)) {{ }} Calling function func_name((value ,, value))	This the syntax of a function Ex: >IntegeR< sum ((>IntegeR< a,, >IntegeR< b)) {{ >IntegeR< s s := a + b ->OuT<- (("sum")) >BacK< s }} sum((5 ,, 7))
19	ContinuE BreaK	For continuing the loop For breaking from loop
20	IncrementT1 DecrementT1	For incrementing the variable value by one Syntax: variable_name IncremenT1 Ex: num2 IncremenT1 For decrementing the variable value by one Syntax: variable_name DecremenT1 Ex: num2 DecremenT1