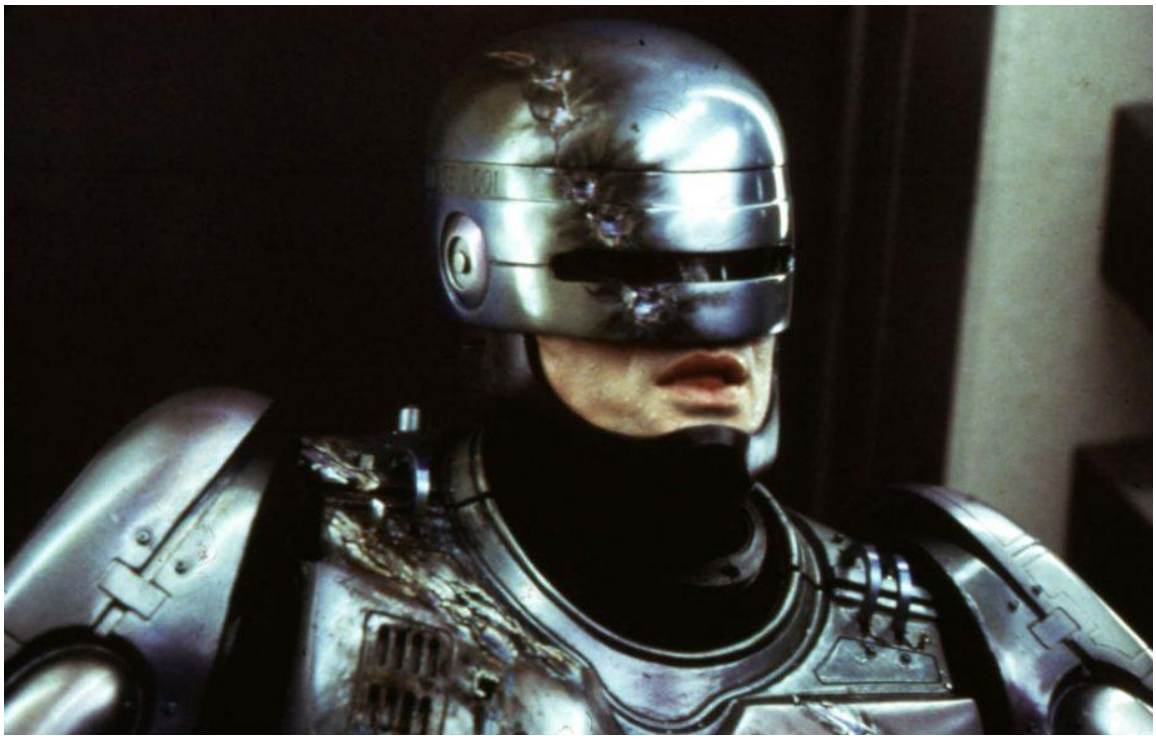
CS315 PROJECT

ROBOCOD LANGUAGE



Group Members:

Asuman Aydın – Section 01 - 21502604

Doruk Altan – Section 02 - 21401362

Fatih Sabri Aktepe – Section 02 - 21000541

As the name itself implies, ROBOCOD is a language that is designed to excel at programming robots. Robots are expected to respond to data from the environment via various sensors or directly from the master. The robot's main functions are to move, change direction, grab and release objects so when naming the reserved words we tried to be as literal as we can so that anyone can easily understand the language. Whether writing or reading, we trust out language is easy to understand. Aside from the features that are aimed to increase efficiency in programming robots specifically, our language shows similarities to the C++ language which we have the most experience with, therefore many of our constructs which already exist in many other languages are closer to C++. A user with just a little familiarity with C languages should be able to jump right in ROBOCOD.

PART A:

<program> → <stmts>

<stmts> → <stmts>

| <functions>; <stmts>

| <stmt> ; <stmts>

| <empty>

<stmt> → <assignment_expr>

| <logic_expression>

| <turning>

|// <comment>//

|<empty>

| <if_stmt>

<functions> → <function>

| <function> <function>

<function> → <identifier> (<parameter list>) {<body>}

<body> → <stmts>

<move> → moveTheRobot(<parameter list>)

<turn> → turnTheRobot(<parameter list>)

<grab> → grabObject(<parameter list>)

<release> → releaseObject(<parameter list>)

<scan>→ scan(<parameter list>)

<send> → sendData(<parameter list>)

<receive> → getData(<parameter list>)

<parameter list> → <empty>

|<parameter>

|<parameter>, <parameter list>

<parameter> → <value>

| <logic>

<if_stmt> → <matched>

| <unmatched>

<matched> → if (<logic_expression >) <matched> else <matched>

| <assignment_exp>

<unmatched> → if (<logic_expression >) [<stmts>]

| if (<logic_expression >) <matched> else <unmatched> [<stmts>]

<logic_expression> → <logic> = <logic>

<logic> → true

| false

<assignment_exp> → <value> = < arithmetic_op >

| <value> = <identifier>

< value > → robot | number| …

<identifier> → <value> | <moveTerm> | <int> | <string> | <logic_expression>

<int> → <digit><int> | <digit>

<string> → <letters>

<letters> → a|b|….z|A| B| C….Z

<digit> → 0|1|2|3|…|9

<float> → <digit><integers>.<digit><integers>

< arithmetic_op > → <moveTerm> + < moveTerm >

| < moveTerm > - < moveTerm >

| < moveTerm > / < moveTerm >

| < moveTerm > * < moveTerm >

< moveTerm > → <value> | const

<turning> → <turning_conditional>

| <turning_round>

<turning_conditional> → while (<logic_expression >) [<stmts>]

<turning_round> → for(<assignment_exp> ; <logic_expression>; <assignmen_exp>)[<stmts>]

<either> → <either> ? <onemore>
| <onemore>
<onemore> → <onemore> ^^ <noway>
| <noway>
<noway> : /\ <term>
| <term>
<term> → <logic> | <value> | ..

<comments> → every ascii except “//”

<empty> →

EXPLANATİON OF BNF :

<program> : It is a starter for the program. And it is simply sees the statements which has sub-statement types. After this statements can come and the program starts.

<stmts> : It is the basis of all implementation mostly. It can be function declaration, assignment statement or a empty line. Statements do not need any sign in the end to finish it.

<function> : This language allows user to define his/her own functions with parameters or without. However the arithmetic operations of language are limited, the function will not be advanced more than a robot can do in a basic way.

<body>: A body can contain statements inside a function or for loop for example.

<move> : It is s constant function that every robot in this language can execute with constant number of progress.

<turn> : Like move, this declaration is robot's basic instruction and it has also constant degree to turn as a parameter. The other primitive functions are in order: <grab>,<release> ,<scan>,<send> ,<receive> .

<parameter list> : It is optional for the functions. This description allows user to ease its job between the functions and statements. It can be a value or logic value.

<matched>: It basically sets precedence over if and else conditional statements.

<**unmatched>** : this symbolize the relation between multiple if-else statements.

<logic_expression> : These expressions can have true or false logical value.

<assignment_exp> : It is for declaration and to check logical equivalence.

<value> : It is for declarations of assignment operators.

<identifier> : This non-terminal contains any value such as string, numbers and etc.

<int> : it is any number between 0 and infinite as a one digit.

<string> : To state a message or any other input data, this non-terminal is used.

<letters> : For the characters of a string.

<digit> : This non-terminal is used for the numbers from 0-9.

<float> : It is for the functions to user milimetric calculations.

< arithmetic_op > : four basic calculation is controlled by this non-terminal.

< moveTerm > : For the calculations, defined values in language.

<turning> : It is loop statements. The word choice of turning is inspired by robot's moves.

<either> : It is for OR logical operator.
<onemore> : It is for AND logical operator.
<noway> : It basically says, there cannot be such a thing so it is NOT logical operator.
<comments> : For better use of the language, user can write something other than the code that will not be executed.
<empty>: it is an empty statement, can be a new line basically or emptiness of parameter in the function.

PART B: -LEX DESCRİPTİON

```
%option main

integer [0-9]+

letters [a-zA-Z]+

float [0-9]+\.[0-9]+

%%

scan                printf("SCAN");

print               printf("PRINT");

moveTheRobot            printf("MOVING_FUNC");

turnTheRobot        printf("TURNING_FUNC");

grabObject          printf("GRAB_FUNC");

releaseObject       printf("RELEASE_FUNC");

letter              printf("LETTER");

string              printf("STRING");

digit               printf("DIGIT");

integer             printf("INTEGER");

float               printf("FLOAT");

if                  printf("IF");

else                printf("ELSE");

while               printf("WHILE");

for                 printf("FOR");
```

```
true                    printf("BOOL_TRUE");
false                   printf("BOOL_FALSE");
\+                      printf("ADDITION_OP");
\-                      printf("SUBSTRACTION_OP");
\*                      printf("MULTIPLICATION_OP");
\/                      printf("DIVISION_OP");
\^^                     printf("AND_OP");
\?                      printf("OR_OP");
\/\                     printf("NEGATION");
\=                      printf("ASSIGNMENT_OP");
\;                      printf("SEMI_COLON");
\,                      printf("COMMA");
\(                      printf("LP");
\)                      printf("RP");
\{                      printf("L_CURLY");
\}                      printf("R_CURLY");
\<                      printf("SMALLER_OP");
\>                      printf("GREATER_OP");
\[                      printf("FUNCT_DERİVER");
\]                      printf("FUNCT_RECEİVER");
\"                      printf("STRING_DEFINER");
[a-zA-Z][_a-zA-Z0-0]*   printf("IDENTIFIER");
```

As it is written in the assignment, this is a robotic language. However this language is more like for dummy robots. It cannot do much but it can survive in nature. In life, even we humans have basic properties to execute to survive and when we consider the environment, life becomes easier with that. These language responses to these people who want a basic language to implement their basic instructions also. So readability wise, the ROBOCOD language is a basic language for now that is simple to read like reading a children's book. We may need to work on the writability part but for now; it executes what a robot wants.

PART C:

```
String robot = "Life is good"
String value = "Obje to grab"
moveTheRobot (int steps){ steps = steps +1}
 turnTheRobot(int degree){ degree = degree + 1}
 grabObject(string objectMessage){}
releaseObject(float coordinates){}
scan(string inputData){}
sendData(){}
 getData(int data){}
int value = 5;
for(int i = 0; i< 10; i = i + 1)[moveTheRobot(value)]
if(value == 15)[scan(robot)]
else[grabObject(value) ]
```