Quick - generare de cod

```
program ::= {
        // codul de la generarea de cod se adauga dupa codul celorlalte componente
        crtCod=&tMain;
        crtVar=&tInceput;
        Text_write(&tInceput,"#include \"quick.h\"\n\n");
        Text_write(&tMain,"\nint main(){\n");
        }
        ( defVar | defFunc | block )* FINISH
        {
        Text_write(&tMain,"return 0;\n}\n");
        FILE *fis=fopen("1.c","w");
        if(!fis){
                printf("cannot write to file 1.c\n");
                exit(EXIT_FAILURE);
                }
        fwrite(tInceput.buf,sizeof(char),tInceput.n,fis);
        fwrite(tFunctii.buf,sizeof(char),tFunctii.n,fis);
        fwrite(tMain.buf,sizeof(char),tMain.n,fis);
        fclose(fis);
        }
defVar ::= VAR ID COLON baseType SEMICOLON
        {
        Text_write(crtVar,"%s %s;\n",cType(ret.tip),nume);
        }
defFunc ::= FUNCTION ID
        {
        crtCod=&tFunctii;
        crtVar=&tFunctii;
        Text_clear(&tAntetFn);
        Text_write(&tAntetFn,"%s(",nume);
        }
        LPAR funcParams RPAR COLON baseType
                {
                Text_write(&tFunctii,"\n%s %s){\n",cType(ret.tip),tAntetFn.buf);
                }
                defVar* block END
                        {
                        Text_write(&tFunctii,"}\n");
                        crtCod=&tMain;
                        crtVar=&tInceput;
                        }
funcParams ::= ( funcParam ( COMMA
        {
        Text_write(&tAntetFn,",");
        }
        funcParam )* )?
funcParam ::= ID COLON baseType
        {
        Text_write(&tAntetFn,"%s %s",cType(ret.tip),nume);
        }
instr ::= expr? SEMICOLON
                {
```

```
                Text_write(crtCod,";\n");
                }
          | IF LPAR
                {
                Text_write(crtCod,"if(");
                }
                expr RPAR
                      {
                      Text_write(crtCod,"){\n");
                      }
                      block
                            {
                            Text_write(crtCod,"}\n");
                            }
                            ( ELSE
                                  {
                                  Text_write(crtCod,"else{\n");
                                  }
                                  block
                                        {
                                        Text_write(crtCod,"}\n");
                                        }
                                        )? END
          | RETURN
                {
                Text_write(crtCod,"return ");
                }
                expr SEMICOLON
                      {
                      Text_write(crtCod,";\n");
                      }
          | WHILE
                {
                Text_write(crtCod,"while(");
                }
                LPAR expr RPAR
                      {
                      Text_write(crtCod,"){\n");
                      }
                      block END
                      {
                      Text_write(crtCod,"}\n");
                      }
exprLogic ::= exprAssign ( ( AND
          {
          Text_write(crtCod,"&&");
          }
          | OR
          {
          Text_write(crtCod,"||");
          }
          ) exprAssign )*
exprAssign ::= ( ID ASSIGN
          {
          Text_write(crtCod,"%s=",nume);
          }
```

```
        )? exprComp
exprComp ::= exprAdd ( ( LESS
        {
        Text_write(crtCod,"<");
        }
        | EQUAL
        {
        Text_write(crtCod,"==");
        }
        ) exprAdd )?
exprAdd ::= exprMul ( ( ADD
        {
        Text_write(crtCod,"+");
        }
        | SUB
        {
        Text_write(crtCod,"-");
        }
        ) exprMul )*
exprMul ::= exprPrefix ( ( MUL
        {
        Text_write(crtCod,"*");
        }
        | DIV
        {
        Text_write(crtCod,"/");
        }
        ) exprPrefix )*
exprPrefix ::= ( SUB
        {
        Text_write(crtCod,"-");
        }
        | NOT
        {
        Text_write(crtCod,"!");
        }
        )? factor
factor ::= INT
        {
        Text_write(crtCod,"%d",consumed->i);
        }
        | REAL
        {
        Text_write(crtCod,"%g",consumed->r);
        }
        | STR
        {
        Text_write(crtCod,"\"%s\"",consumed->s);
        }
        | LPAR
        {
        Text_write(crtCod,"(");
        }
        expr RPAR
                {
                Text_write(crtCod,")");
```

```
          }
| ID
{
Text_write(crtCod,"%s",s->nume);
}
( LPAR
          {
          Text_write(crtCod,"(");
          }
          ( expr ( COMMA
                    {
                    Text_write(crtCod,",");
                    }
                    expr )* )? RPAR
                              {
                              Text_write(crtCod,")");
                              }
                              )?
```