Quick - generare de cod

```
program ::= {
        // the code for code generation will be added after the code of the other modules
        crtCode=&tMain;
        crtVar=&tBegin;
        Text_write(&tBegin,"#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(tBegin.buf,sizeof(char),tBegin.n,fis);
        fwrite(tFunctions.buf,sizeof(char),tFunctions.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.type),name);
        }
defFunc ::= FUNCTION ID
        {
        crtCode=&tFunctions;
        crtVar=&tFunctions;
        Text_clear(&tFnHeader);
        Text_write(&tFnHeader,"%s(",name);
        }
        LPAR funcParams RPAR COLON baseType
                {
                Text_write(&tFunctions,"\n%s %s){\n",cType(ret.type),tFnHeader.buf);
                }
                defVar* block END
                        {
                        Text_write(&tFunctions,"}\n");
                        crtCode=&tMain;
                        crtVar=&tBegin;
                        }
funcParams ::= ( funcParam ( COMMA
        {
        Text_write(&tFnHeader,",");
        }
        funcParam )* )?
funcParam ::= ID COLON baseType
        {
        Text_write(&tFnHeader,"%s %s",cType(ret.type),name);
        }
instr ::= expr? SEMICOLON
                {
```

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

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

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