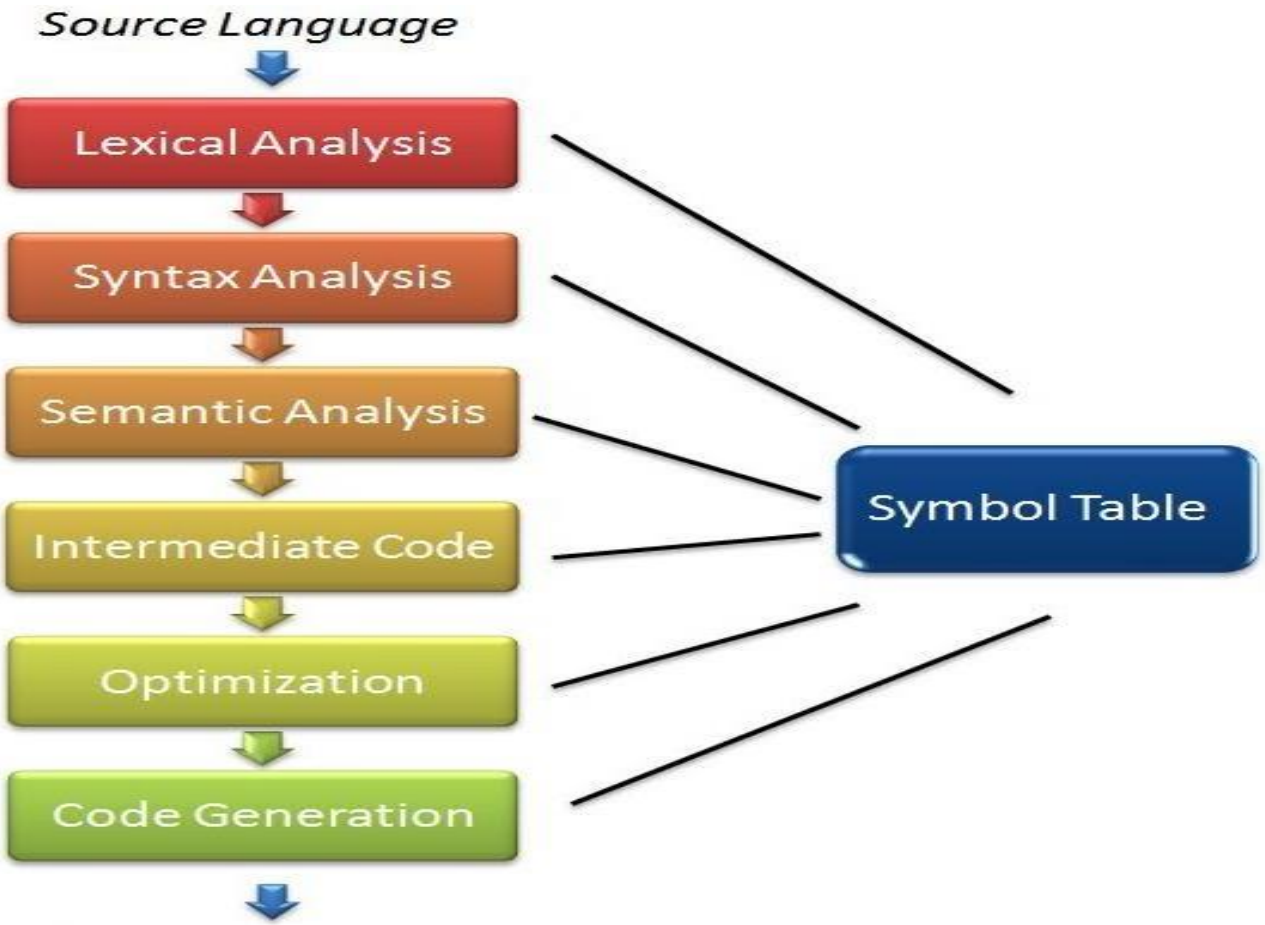
Symbol Table / טבלת הסמלים



Symbol Table: This part of compiler is responsible for storing **all the Type Information** about all variables or functions.

טבלת הסמלים / Symbol Table

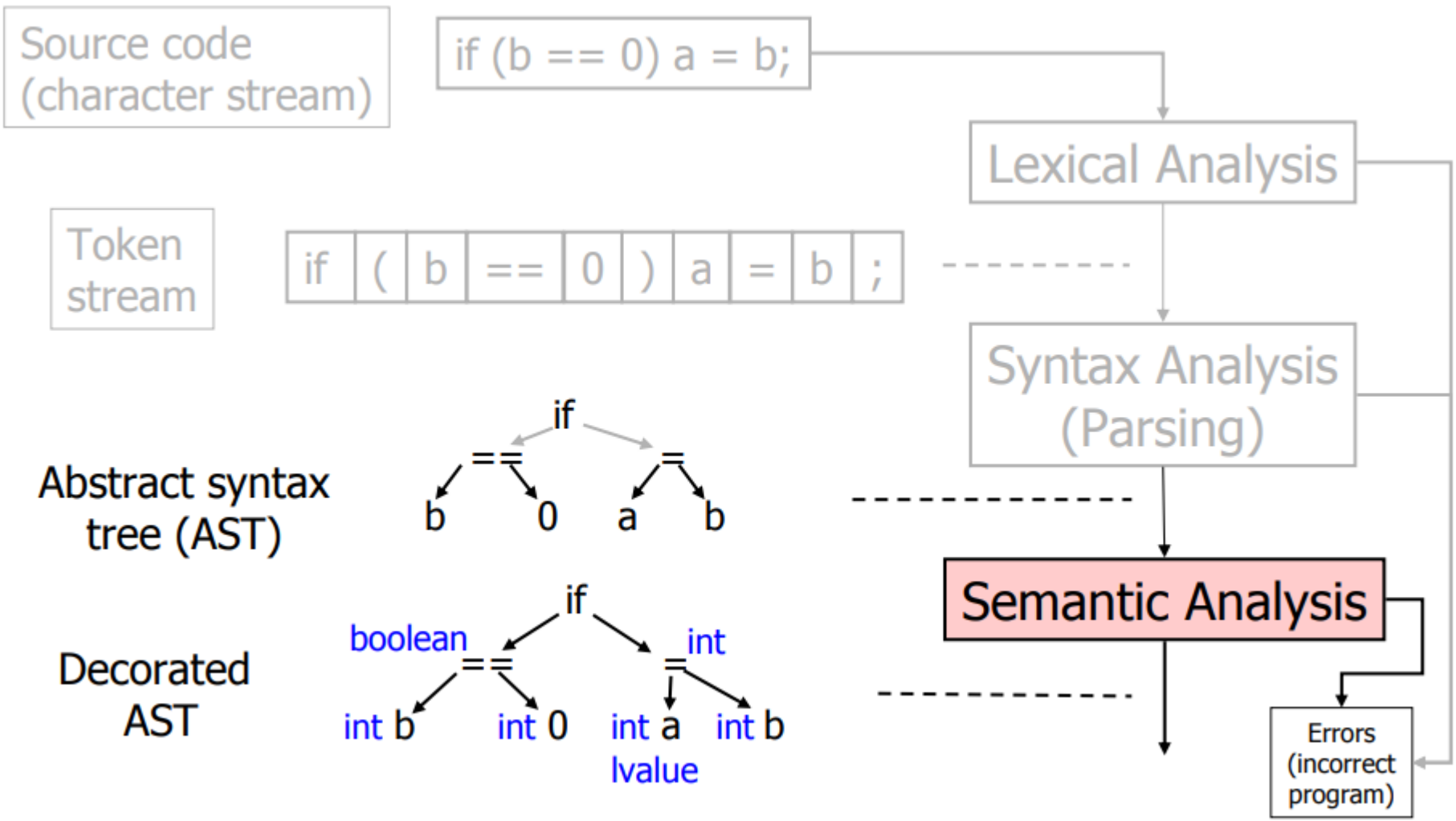
ניתוח סמנטי

- **המנתח הסמנטי** הוא מנצל את הערכים הסמנטיים של הטרמינלים (אסימונים) ואת המבנה ההיררכי של התוכנית (העץ שחושב על ידי הניתוח התחבירי).

- קלט: עץ הגזירה של התוכנית (מהמנתח התחבירי), והערכים הסמנטיים של האסימונים (מהמנתח הלקסיקלי).

- פלט: אם התוכנית חוקית : **עץ גזירה מעוטר / Decorated AST**– עץ המכיל ערכי סמנטיים גם עבור חלקים התחביריים של התוכנית (כלומר – מופעים של משתנים בעץ הגזירה).

Symbol Table / טבלת הסמלים

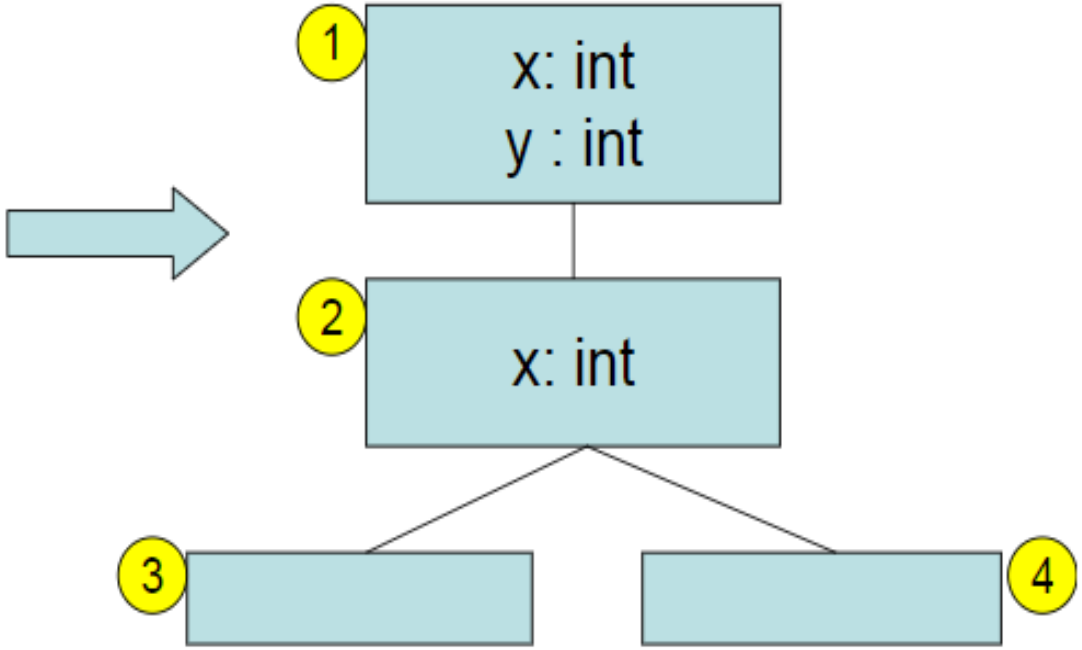


Symbol Table / טבלת הסמלים

רעיון: לכל scope ניצור טבלה נפרדת

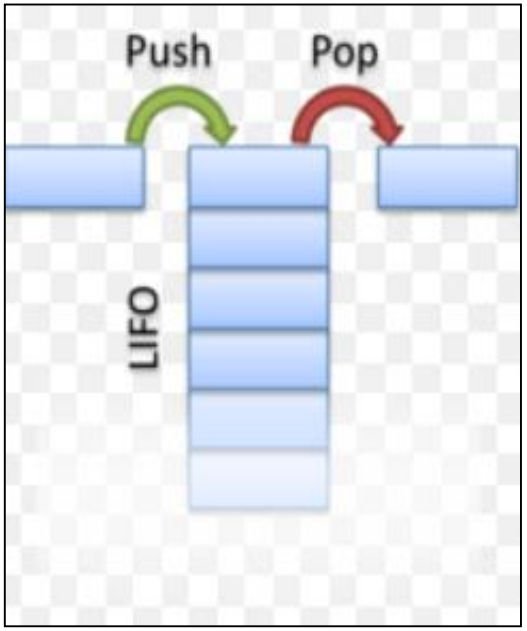
- דוגמה:

```
1 {
    int x,y;
  2 {
        int x;
    3 {
          y = x;
        }
    4 { ... }
    }
}
```



Symbol Table / טבלת הסמלים

- הבנייה מתבצעת <u>בזמן קומפילציה</u>
- נשתמש במחסנית שמחזיקה את שרשרת הטבלאות מה-scope הנוכחי עד לשורש
  - על מנת למצוא משתנה, מספיק לבדוק רק טבלאות במחסנית
  - בכניסה ל-scope ניצור טבלה חדשה ונדחוף למחסנית
  - ביציאה מ-scope נוציא טבלה מראש המחסנית

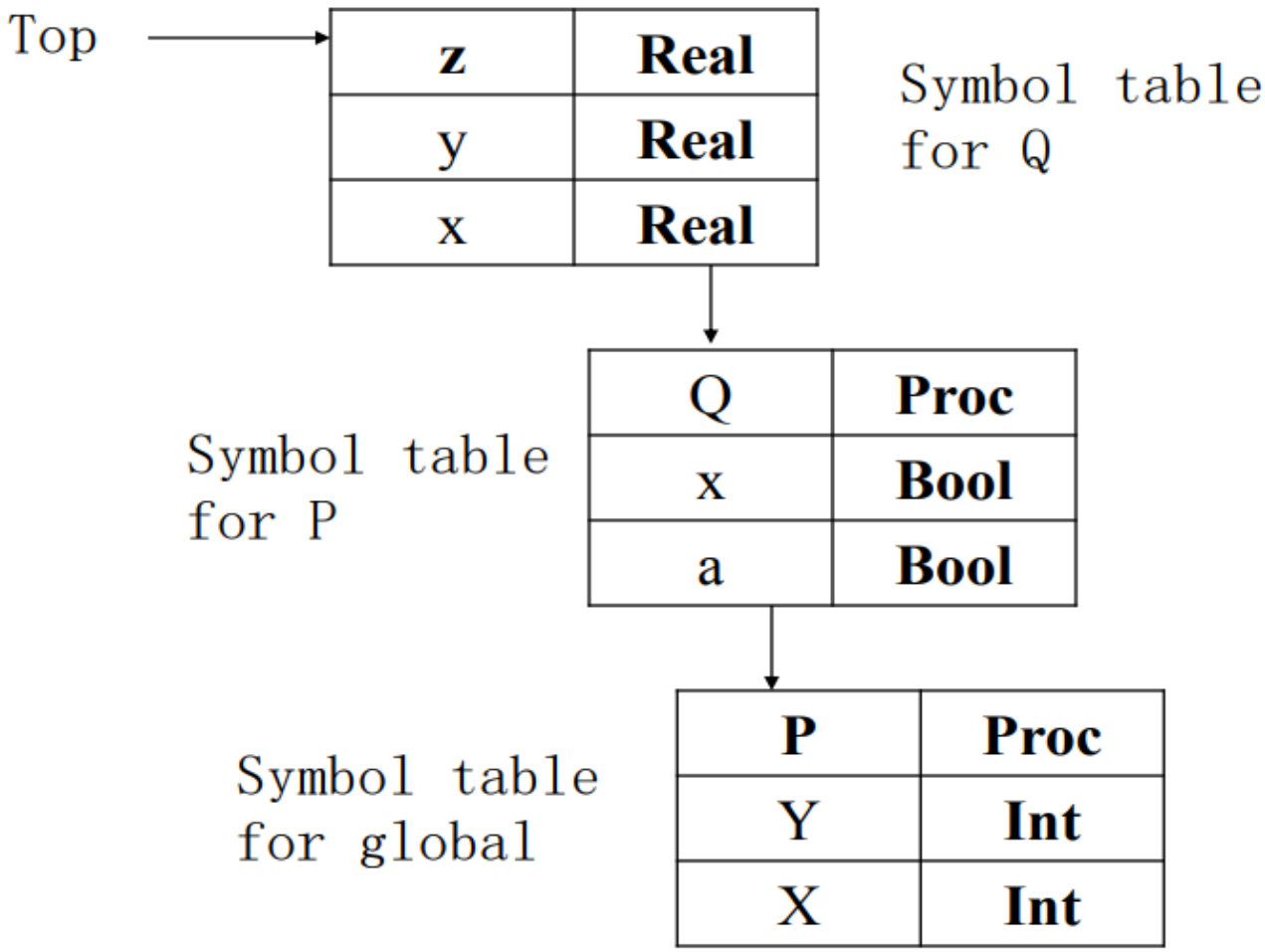


- כדי למצוא משתנה x:
  - חפש בטבלת הסמלים של ה-scope הנוכחי. אם מצאת, עצור. אחרת,
  - חפש בטבלת הסמלים של האבא של ה-scope הנוכחי. אם מצאת, עצור. אחרת,
  - המשך את החיפוש עד לשורש

Symbol Table / טבלת הסמלים

1 SYMBOL TABLE ORGANIZATION

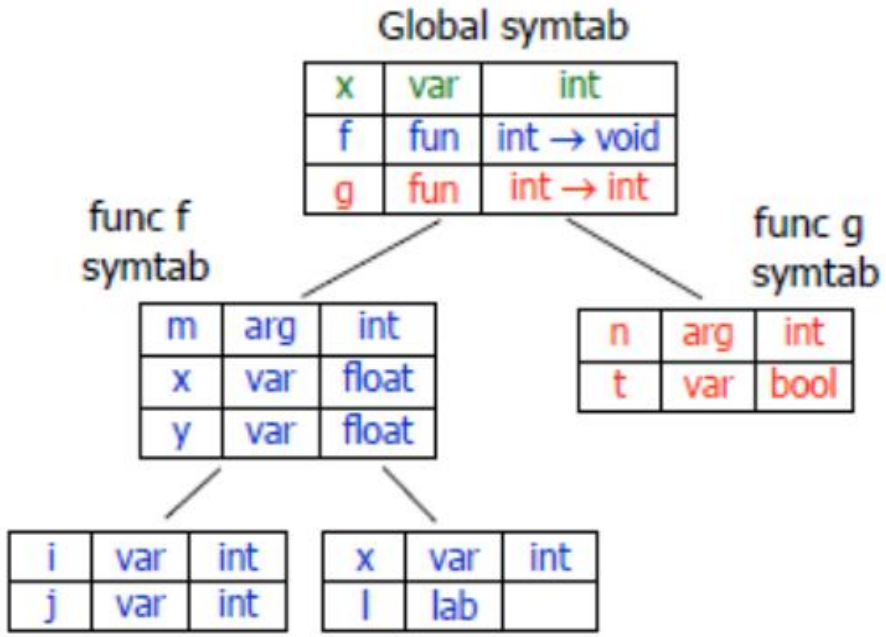
```
Int x,y;

Procedure P:
   Bool x, a ;

    Procedure Q:
       Real x,y,z ;
       begin
       ......
        end
    begin

    end
```



Symbol Table / טבלת הסמלים

2 SYMBOL TABLE ORGANIZATION

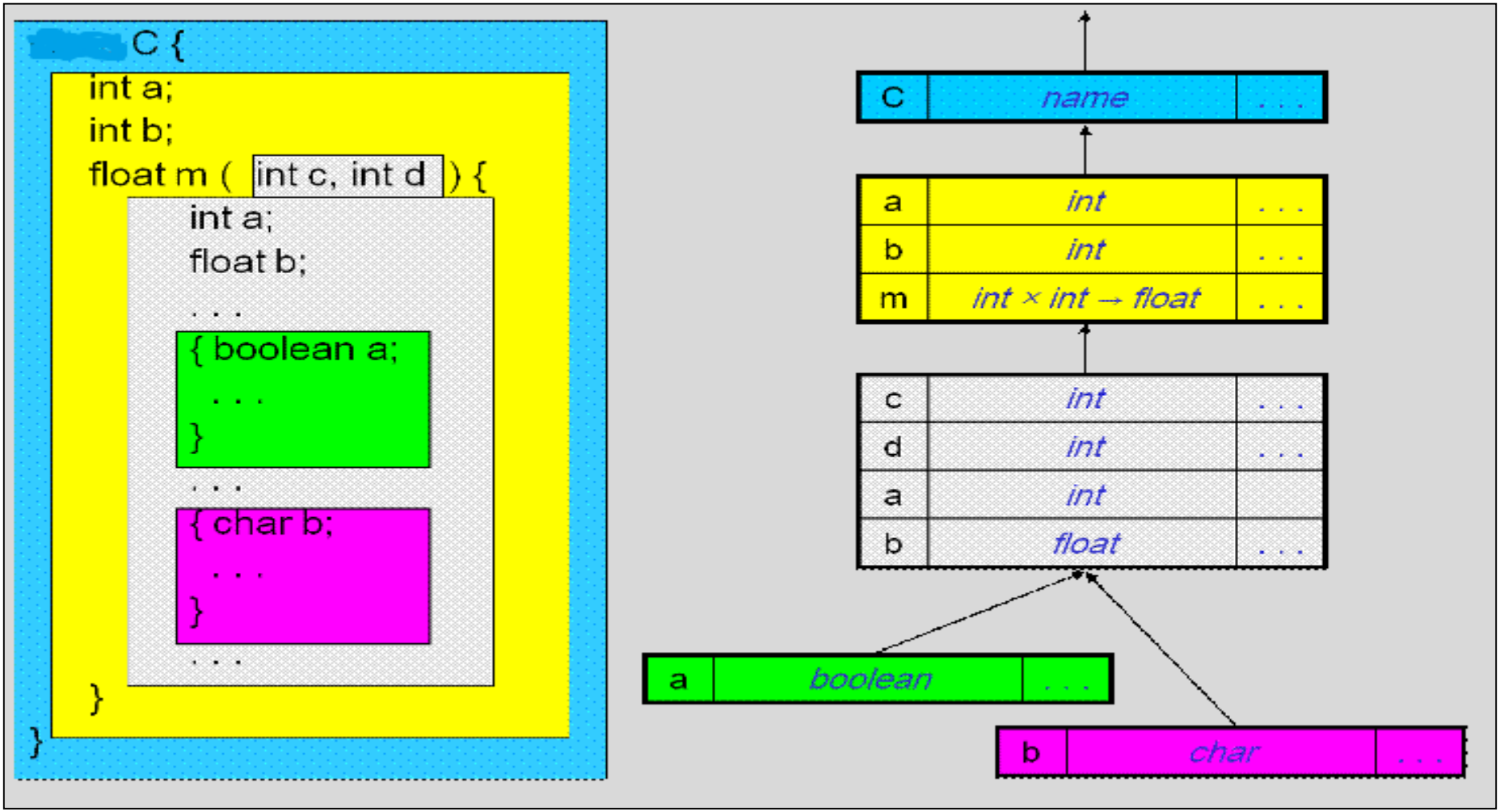
```
int x;

void f(int m) {
    float x, y;
    ...
    { int i, j; ...; }
    { int x; l: ...; }
}

int g(int n) {
    bool t;
    ...;
}
```



- Entries in the symbol table will contain information about every identifier, its type, its position in storage, and any other information relevant to the translation process.

Symbol Table / טבלת הסמלים

3 SYMBOL TABLE ORGANIZATION



Symbol Table / טבלת הסמלים

- Syntactically correct programs may still contain errors !
  - Lexical analysis does not distinguish between different variable names (same ID token)
  - Syntax analysis does not correlate variable declaration with variable use, does not keep track of types

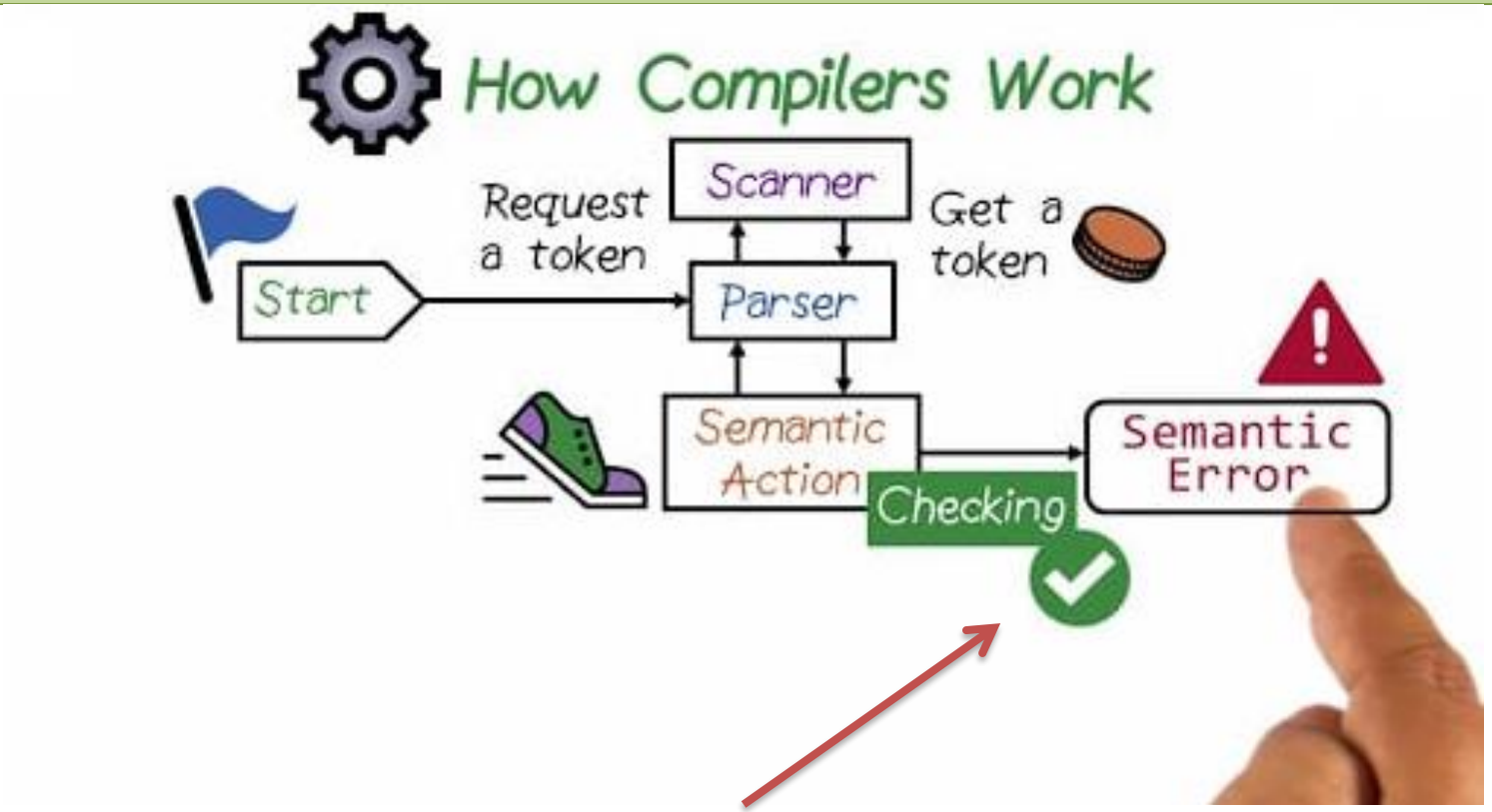
```
int a;
a = "hello";
```

Assigning wrong type

```
int a;
b = 1;
```

Assigning undeclared variable

Symbol Table / טבלת הסמלים



אם התוכנית אינה חוקית: **הודעת שגיאה מתאימה.**

Semantic Errors:

- Type mismatch
- Undeclared variables
- Reserved identifier misuse

Symbol Table / טבלת הסמלים

INPUT / OUTPUT EXAMPLE 1 :

ENTER ID NAME :
a1

int a1;
double b2;
char c3;
int d4;

^D (means end of input stream)

OK

INPUT / OUTPUT EXAMPLE 2 :

ENTER ID NAME :
a1

int a1;
double a1;
char c3;
int d4;

^D (means end of input stream)

OK

REDECLARATION a1

Symbol Table / טבלת הסמלים

INPUT / OUTPUT EXAMPLE 3 :

ENTER ID NAME :
a1

int a1;
double b2;
int a1;
int d4;

^D (means end of input stream)

Semantic Error

OK

DECLARATION ERROR a1

Symbol Table / טבלת הסמלים

LEX file: (tbl.l)

```
%{
 #include <string.h>
%}
%%
int return INT;
double return  DOUBLE;
char return CHAR;
[a-z]+[a-z0-9]* { yylval.s = strdup(yytext); return ID; }
[ \t]  printf(" ");
. return  yytext[0];
.|\n  ;
%%
```

Symbol Table / טבלת הסמלים

YACC file: (tbl.y)

```
%union  {
 char *s;
};
%{
 #include<stdio.h>
 #include <string.h>
 char id_arr[10][10],id[10];
 int type_arr[10],i = 0, j = 0, k = 0;
 int flag = 0, first_type = 0;
%}
%token INT,DOUBLE,CHAR
%token<s>  ID
%%
S: EXP  { printf("OK\n");  print();  test(); }
EXP: EXP DEF | DEF;
DEF: INT ID ';'    { strcpy(id_arr[i],$2); i++; type_arr[j]=0; j++; };
       | DOUBLE  ID ';' { strcpy(id_arr[i],$2); i++; type_arr[j]=1; j++; };
       | CHAR  ID ';' ; { strcpy(id_arr[i],$2); i++; type_arr[j]=2; j++; };
%%
```

Symbol Table / טבלת הסמלים

```
#include "lex.yy.c"
main()
{
  printf("ENTER ID NAME :\n ");
  scanf("%s\n  ",id);
  return  yyparse();
}
int yyerror()
{
  printf("MY error\n");
  return 0;
}
void print()
{
  for(k=0;k<i;k++)
  {
    printf("%d   ",type_arr[k]);
    printf("%s\n",id_arr[k]);
  }
 }
```

Symbol Table / טבלת הסמלים

```
void test()
{
 for(k=0;k<i;k++)
 {
    if(strcmp(id_arr[k],id)==0)
    {
      if(flag==0)
      {
        flag=1;
        first_type=type_arr[k];
      }
      else
      {
        if(first_type==type_arr[k])
           printf("DECLARATION ERROR  %s\n",id);
        else
           printf("REDECLARATION %s\n",id);
      }
    }
  }
}
```

INT	DOUBLE	CHAR
0	1	2

type_arr	id_arr
0	a1
1	b2
2	c3
0	d4

המנתח התחבירי - YACC

nano tbl.l

בניית קובץ LEX

nano tbl.y

בניית קובץ YACC

lex tbl.l

קומפילציה LEX

yacc tbl.y

קומפילציה YACC

cc -o tbl y.tab.c –ll –Ly

קומפילציה C

./tbl

Input statements

הרצת דוגמאות 1 - 3

^D

Symbol Table / טבלת הסמלים

INPUT / OUTPUT EXAMPLE 1 :

ENTER ID NAME :
a1

int a1;
double b2;
char c3;
int d4;

^D (means end of input stream)

OK

type_arr	id_arr
0	a1
1	b2
2	c3
0	d4

INPUT / OUTPUT EXAMPLE 2 :

ENTER ID NAME :
a1

int a1;
double a1;
char c3;
int d4;

^D (means end of input stream)

OK

type_arr	id_arr
0	a1
1	a1
2	a3
1	a4

REDECLARATION a1

Symbol Table / טבלת הסמלים

INPUT / OUTPUT EXAMPLE 3 :

ENTER ID NAME :
a1

int a1;
double b2;
int a1;
int d4;

^D (means end of input stream)

type_arr	id_arr
0	a1
1	b2
0	a1
1	d4

OK

DECLARATION ERROR a1