

Semantic Analysis

הבנת המשמעות של המשפט
הים התיכון קיבל 100 בבגרות באנגלית

חימום

שגיאות?	קוד
	<pre>void main(void) { int i=0; int j=0; printf("",i,j); }</pre>

חימום

שגיאות?	קוד
<pre>Build: 1 succeeded</pre>	<pre>void main(void) { int i=0; int j=0; printf("",i,j); }</pre>

חימום

שגיאות?	קוד
<div data-bbox="285 534 1008 625">Build: 1 succeeded</div>	<pre data-bbox="1290 429 1709 758">void main(void) { int i=0; int j=0; printf("",i,j); }</pre>
	<pre data-bbox="1290 829 1667 1372">class A { public: int x; } void main(void) { A *a; a->y; }</pre>

חימום

שגיאות?	קוד
<pre>Build: 1 succeeded</pre>	<pre>void main(void) { int i=0; int j=0; printf("",i,j); }</pre>
<pre>error C2039: 'y' : is not a member of 'A'</pre>	<pre>class A { public: int x; } void main(void) { A *a; a->y; }</pre>

חימום

שגיאות?	קוד
	<pre>class A { public: int x; }; void main(void) { A *a; a.x; }</pre>

חימום

שגיאות?	קוד
<pre>error C2228: left of '.x' must have class/struct/union</pre>	<pre>class A { public: int x; }; void main(void) { A *a; a.x; }</pre>

חימום

שגיאות?	קוד
	<pre>class A { private: int x; }; void main(void) { A *a; a->x; }</pre>

חימום

שגיאות?	קוד
<pre>error C2248: 'A::x' : cannot access private member declared in class 'A'</pre>	<pre>class A { private: int x; }; void main(void) { A *a; a->x; }</pre>

חימום

שגיאות?	קוד
	<pre>void main(void) { int i=9/0; }</pre>

חימום

שגיאות?	קוד
<pre>error C2124: divide or mod by zero</pre>	<pre>void main(void) { int i=9/0; }</pre>

חימום

שגיאות?	קוד
<pre>error C2124: divide or mod by zero</pre>	<pre>void main(void) { int i=9/0; }</pre>
	<pre>void main(void) { int j=0; int i=9/j; }</pre>

חימום

שגיאות?	קוד
<pre>error C2124: divide or mod by zero</pre>	<pre>void main(void) { int i=9/0; }</pre>
<pre>Build: 1 succeeded</pre>	<pre>void main(void) { int j=0; int i=9/j; }</pre>

חימום

שגיאות?	קוד
	<pre>void main(void) { FILE *f1; while (f1 = 5); }</pre>

חימום

שגיאות?	קוד
<pre>error C2440: '=' : cannot convert from 'int' to 'FILE *'</pre>	<pre>void main(void) { FILE *f1; while (f1 = 5); }</pre>

חימום

שגיאות?

```
error C2440: '=' :  
cannot convert from 'int' to 'FILE *'
```

קוד

```
void main(void)  
{  
    FILE *f1;  
    while (f1 = 5);  
}
```

```
void main(void)  
{  
    FILE *f1;  
    while (f1 = (FILE *) 5);  
}
```


חימום

שגיאות?

```
error C2440: '=' :  
cannot convert from 'int' to 'FILE *'
```

```
Build: 1 succeeded
```

קוד

```
void main(void)  
{  
    FILE *f1;  
    while (f1 = 5);  
}
```

```
void main(void)  
{  
    FILE *f1;  
    while (f1 = (FILE *) 5);  
}
```

חימום

שגיאות?	קוד
	<pre>void main(void) { 13 < "13"; }</pre>

חימום

שגיאות?	קוד
<pre>error C2446: '<' : no conversion from 'const char *' to 'int' warning C4552: '<' : operator has no effect; expected operator with side-effect</pre>	<pre>void main(void) { 13 < "13"; }</pre>

חימום

שגיאות?	קוד
<pre>error C2446: '<' : no conversion from 'const char *' to 'int' warning C4552: '<' : operator has no effect; expected operator with side-effect</pre>	<pre>void main(void) { 13 < "13"; }</pre>
	<pre>void main(void) { "12" < "13"; }</pre>

חימום

שגיאות?	קוד
<pre>error C2446: '<' : no conversion from 'const char *' to 'int' warning C4552: '<' : operator has no effect; expected operator with side-effect</pre>	<pre>void main(void) { 13 < "13"; }</pre>
<pre>Build: 1 succeeded</pre>	<pre>void main(void) { "12" < "13"; }</pre>

חימום

שגיאות?	קוד
	<pre>void main(void) { (void) printf("Moish\n"); }</pre>

חימום

שגיאות?	קוד
<div data-bbox="285 549 1010 642">Build: 1 succeeded</div>	<div data-bbox="1232 471 1825 685"><pre>void main(void) { (void) printf("Moish\n"); }</pre></div>

חימום

שגיאות?	קוד
<div data-bbox="285 549 1010 642">Build: 1 succeeded</div>	<pre data-bbox="1232 471 1825 682">void main(void) { (void) printf("Moish\n"); }</pre>
	<pre data-bbox="1232 735 1825 1049">typedef struct { int x; int y; struct PointType NextPoint; } PointType;</pre>

חימום

שגיאות?	קוד
<pre>Build: 1 succeeded</pre>	<pre>void main(void) { (void) printf("Moish\n"); }</pre>
<pre>error C2079: '<unnamed-tag>::NextPoint' uses undefined struct 'PointType'</pre>	<pre>typedef struct { int x; int y; struct PointType NextPoint; } PointType;</pre>

חימום

שגיאות?	קוד
<pre>Build: 1 succeeded</pre>	<pre>void main(void) { (void) printf("Moish\n"); }</pre>
<pre>error C2079: '<unnamed-tag>::NextPoint' uses undefined struct 'PointType'</pre>	<pre>typedef struct { int x; int y; struct PointType NextPoint; } PointType;</pre>
	<pre>typedef struct { int x; int y; struct PointType *NextPoint; } PointType;</pre>

חימום

שגיאות?	קוד
Build: 1 succeeded	<pre>void main(void) { (void) printf("Moish\n"); }</pre>
error C2079: '<unnamed-tag>::NextPoint' uses undefined struct 'PointType'	<pre>typedef struct { int x; int y; struct PointType NextPoint; } PointType;</pre>
Build: 1 succeeded	<pre>typedef struct { int x; int y; struct PointType *NextPoint; } PointType;</pre>

חימום

שגיאות?	קוד
	<pre>int f(void) { return "340"; }</pre>

חימום

שגיאות?	קוד
<pre>error C2440: 'return' : cannot convert from 'const char [4]' to 'int'</pre>	<pre>int f(void) { return "340"; }</pre>

חימום

שגיאות?	קוד
<pre>error C2440: 'return' : cannot convert from 'const char [4]' to 'int'</pre>	<pre>int f(void) { return "340"; }</pre>
	<pre>int g() { return f(); } int f(void) { return 0; }</pre>

חימום

שגיאות?	קוד
<pre>error C2440: 'return' : cannot convert from 'const char [4]' to 'int'</pre>	<pre>int f(void) { return "340"; }</pre>
<pre>error C3861: 'f': identifier not found</pre>	<pre>int g() { return f(); } int f(void) { return 0; }</pre>

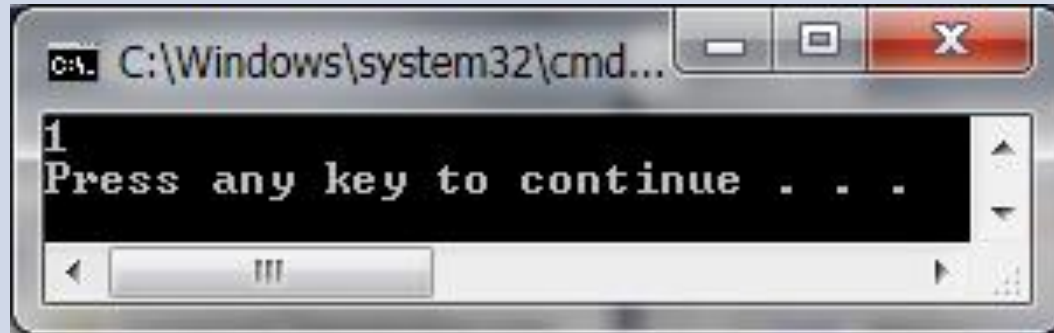
הרצה?

קוד

```
int main(void)
{
    int i=1;
    int j=2;
    int a=3;

    if ((a=i) || (a=j))
    {
        printf("%d\n",a);
    }
}
```


הרצה?

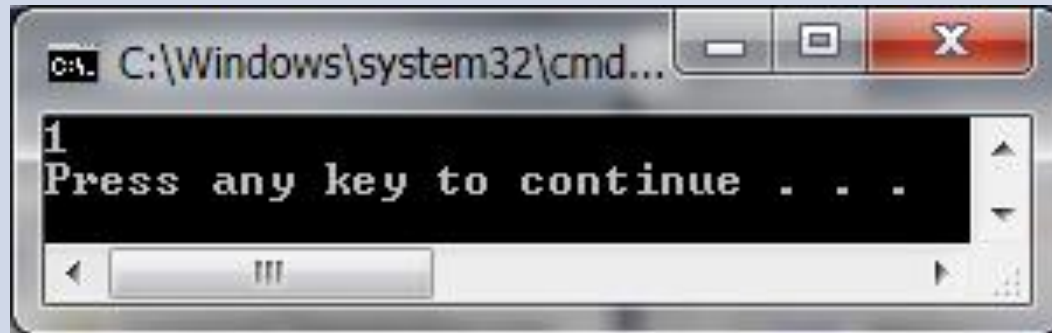


קוד

```
int main(void)
{
    int i=1;
    int j=2;
    int a=3;

    if ((a=i) || (a=j))
    {
        printf("%d\n",a);
    }
}
```

הרצה?



קוד

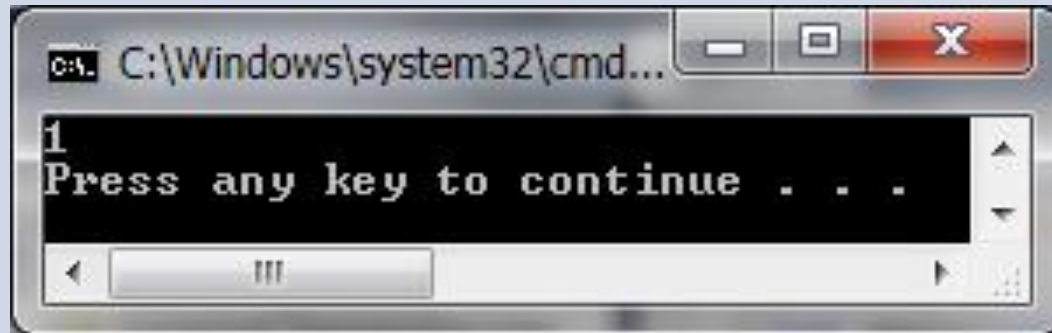
```
int main(void)
{
    int i=1;
    int j=2;
    int a=3;

    if ((a=i) || (a=j))
    {
        printf("%d\n",a);
    }
}
```

```
int main(void)
{
    int i=1;
    int j=2;
    int a=3;

    if ((a=i) | (a=j))
    {
        printf("%d\n",a);
    }
}
```

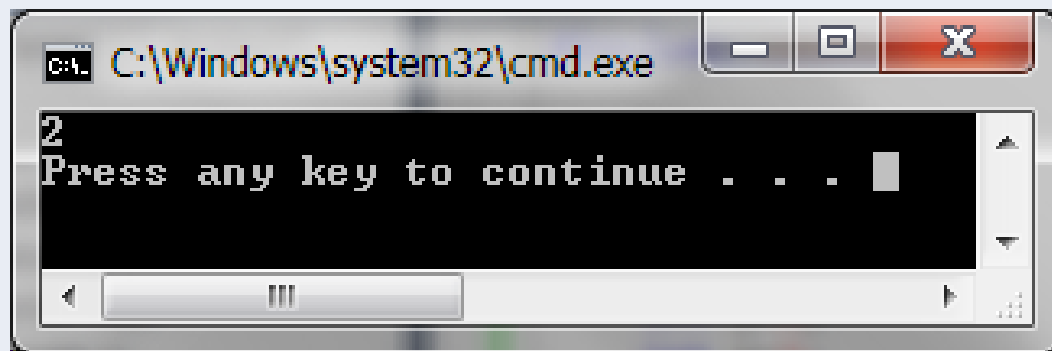
הרצה?



קוד

```
int main(void)
{
    int i=1;
    int j=2;
    int a=3;

    if ((a=i) || (a=j))
    {
        printf("%d\n",a);
    }
}
```



```
int main(void)
{
    int i=1;
    int j=2;
    int a=3;

    if ((a=i) | (a=j))
    {
        printf("%d\n",a);
    }
}
```

הרצה?

קוד

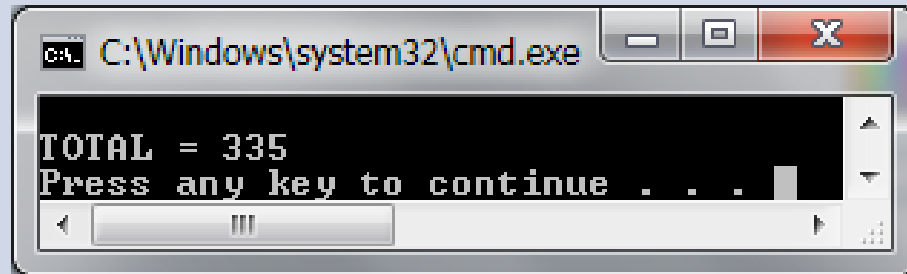
```
srand((unsigned int)time(NULL));

for (i = 0; i < 1000; i++)
{
    double x = cos((double) rand());
    double y = cos((double) rand());
    double z = cos((double) rand());

    if ((x*y)*z != x*(y*z))
    {
        total++;
        printf("%d\n",i);
    }
}

printf("\n\nTOTAL = %d\n", total);
```

הרצה?



קוד

```
srand((unsigned int)time(NULL));

for (i = 0; i < 1000; i++)
{
    double x = cos((double) rand());
    double y = cos((double) rand());
    double z = cos((double) rand());

    if ((x*y)*z != x*(y*z))
    {
        total++;
        printf("%d\n",i);
    }
}

printf("\n\nTOTAL = %d\n", total);
```

הרצה?

קוד

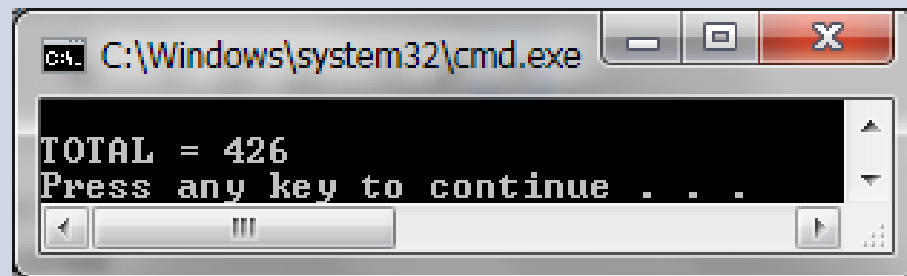
```
srand((unsigned int)time(NULL));

for (i = 0; i < 1000; i++)
{
    double x = cos((double) rand());
    double y = cos((double) rand());

    if ((x+x*y) != (x*(1+y)))
    {
        total++;
        printf("%d\n",i);
    }
}

printf("\n\nTOTAL = %d\n", total);
```

הרצה?



קוד

```
srand((unsigned int)time(NULL));

for (i = 0; i < 1000; i++)
{
    double x = cos((double) rand());
    double y = cos((double) rand());

    if ((x+x*y) != (x*(1+y)))
    {
        total++;
        printf("%d\n",i);
    }
}

printf("\n\nTOTAL = %d\n", total);
```

חימום

שגיאות?

קוד

```
class Father {  
public:  
  
    virtual void SALARY(void){printf("100\n");}  
};  
  
class Son : public Father {  
public:  
  
    virtual void SALARY(void){printf("50\n");}  
};  
  
int main(void)  
{  
    Son *x = new Father;  
  
    x->SALARY();  
}
```


חימום

שגיאות?

קוד

error C2440:

```
class Father {  
public:  
  
    virtual void SALARY(void){printf("100\n");}  
  
};  
  
class Son : public Father {  
public:  
  
    virtual void SALARY(void){printf("50\n");}  
  
};  
  
int main(void)  
{  
    Son *x = new Father;  
  
    x->SALARY();  
}
```

חימום

שגיאות?

קוד

```
class Father {
public:

    virtual void SALARY(void){printf("100\n");}
};

class Son : public Father {
public:

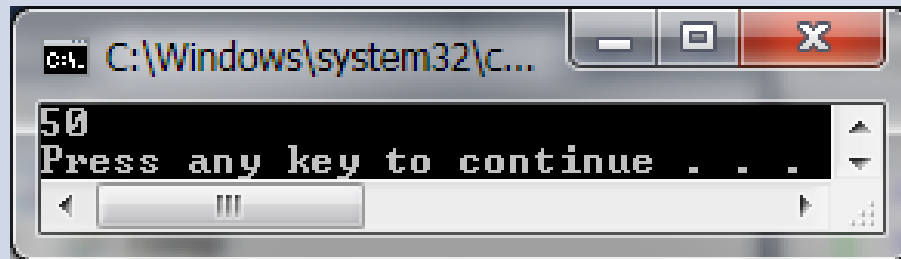
    virtual void SALARY(void){printf("50\n");}
};

int main(void)
{
    Father *x = new Son;

    x->SALARY();
}
```

חימום

שגיאות?



קוד

```
class Father {
public:

    virtual void SALARY(void){printf("100\n");}

};

class Son : public Father {
public:

    virtual void SALARY(void){printf("50\n");}

};

int main(void)
{
    Father *x = new Son;

    x->SALARY();
}
```

חימום

שגיאות?

קוד

```
class Father {
public:

    void SALARY(void){printf("100\n");}

};

class Son : public Father {
public:

    void SALARY(void){printf("50\n");}

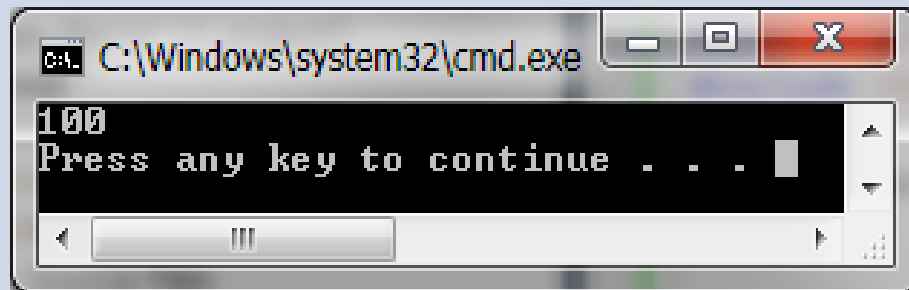
};

int main(void)
{
    Father *x = new Son;

    x->SALARY();
}
```

חימום

שגיאות?



קוד

```
class Father {  
public:  
  
    void SALARY(void){printf("100\n");}  
};  
  
class Son : public Father {  
public:  
  
    void SALARY(void){printf("50\n");}  
};  
  
int main(void)  
{  
    Father *x = new Son;  
  
    x->SALARY();  
}
```

חימום

שגיאות?

קוד

```
class Father {
public:

    virtual void SALARY(void){printf("100\n");}

};

class Son : public Father {
public:

    virtual void SALARY(void){printf("50\n");}
    virtual void SWIM(void){printf("###\n");}

};

void f(Father *f)
{
    f->SWIM();
}

int main(void)
{
    f(new Son);
}
```

חימום

שגיאות?

קוד

```
error C2039: 'SWIM' :  
is not a member of 'Father'
```

```
class Father {  
public:  
  
    virtual void SALARY(void){printf("100\n");}  
};  
  
class Son : public Father {  
public:  
  
    virtual void SALARY(void){printf("50\n");}  
    virtual void SWIM(void){printf("###\n");}  
};  
  
void f(Father *f)  
{  
    f->SWIM();  
}  
  
int main(void)  
{  
    f(new Son);  
}
```

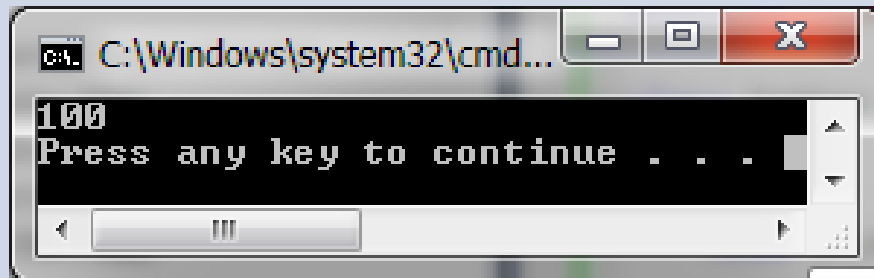
חימום

שגיאות?	קוד
	<pre>class Father { public: void SALARY(void){printf("100\n");} }; class Son : public Father { public: virtual void SALARY(){printf("50\n");} }; int main(void) { Father *x = new Son; x->SALARY(); }</pre>

חימום

שגיאות?

קוד



```
class Father {  
public:  
  
    void SALARY(void){printf("100\n");}  
  
};  
  
class Son : public Father {  
public:  
  
    virtual void SALARY(){printf("50\n");}  
  
};  
  
int main(void)  
{  
    Father *x = new Son;  
  
    x->SALARY();  
}
```

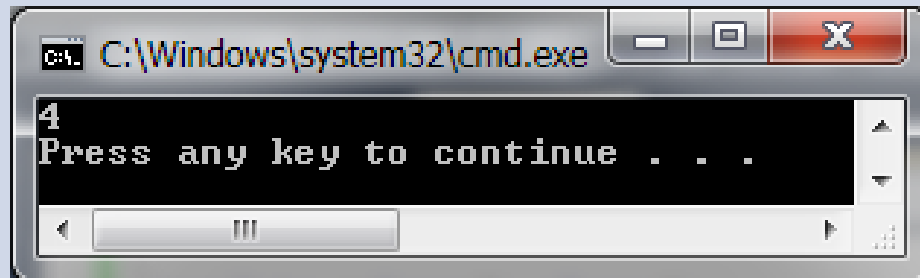
האם באמת רק צ'אק נוריס יכול לשנות ערך של const?

שגיאות?	קוד
	<pre>int main(void) { const int c = 4; int *y = (int *) ((int) &c); *y = 800; printf("%d\n",c); }</pre>

חימום

שגיאות?

קוד



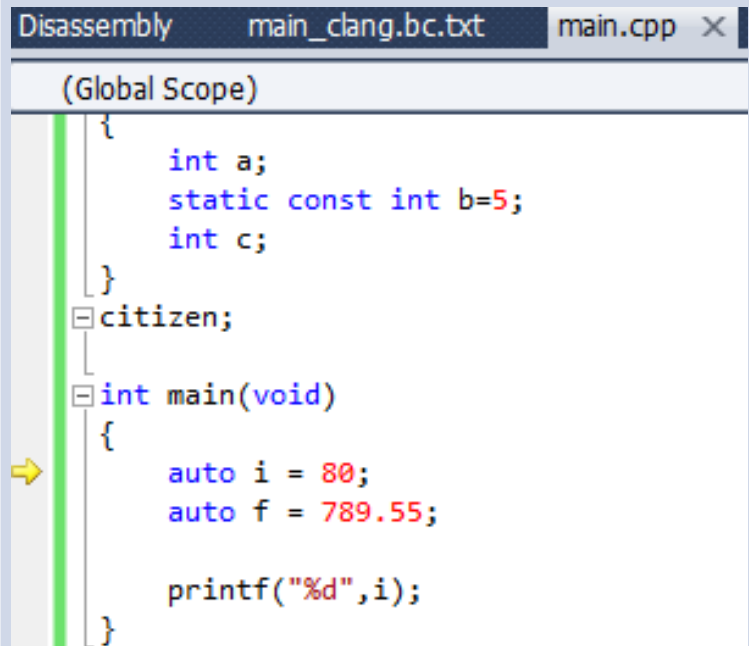
```
int main(void)
{
    const int c = 4;
    int *y = (int *) ((int) &c);
    *y = 800;

    printf("%d\n",c);
}
```

חימום

שגיאות?

קוד



The screenshot shows a code editor with two tabs: 'Disassembly' and 'main_clang.bc.txt'. The 'main.cpp' tab is active, showing the following code:

```
(Global Scope)
{
    int a;
    static const int b=5;
    int c;
}
citizen;
int main(void)
{
    auto i = 80;
    auto f = 789.55;

    printf("%d",i);
}
```

A yellow arrow points to the opening curly brace of the `main` function.

חימום

שגיאות?

Watch 1

Name	Value	Type
i	3931836	int
f	6.356563620102e-317#DEN	double

קוד

Disassembly main_clang.bc.txt main.cpp

(Global Scope)

```
{
    int a;
    static const int b=5;
    int c;
}
citizen;
int main(void)
{
    auto i = 80;
    auto f = 789.55;

    printf("%d",i);
}
```

חימום

שגיאות?	קוד
	<pre>int main(int argc, char **argv) { if (argc > 2) main(2, argv); return 0; }</pre>

חימום

שגיאות?	קוד
<div data-bbox="266 772 879 952">1 succeeded.</div>	<pre data-bbox="1072 649 1831 978">int main(int argc, char **argv) { if (argc > 2) main(2, argv); return 0; }</pre>

חימום

שגיאות?	קוד
	<pre>void f(char *input) { *input = 'G'; } int main(int argc, char **argv) { const char *p = "party"; f(p); return 0; }</pre>

חימום

שגיאות?	קוד
<pre>warning C4090: 'function': different 'const' qualifiers</pre>	<pre>void f(char *input) { *input = 'G'; } int main(int argc, char **argv) { const char *p = "party"; f(p); return 0; }</pre>

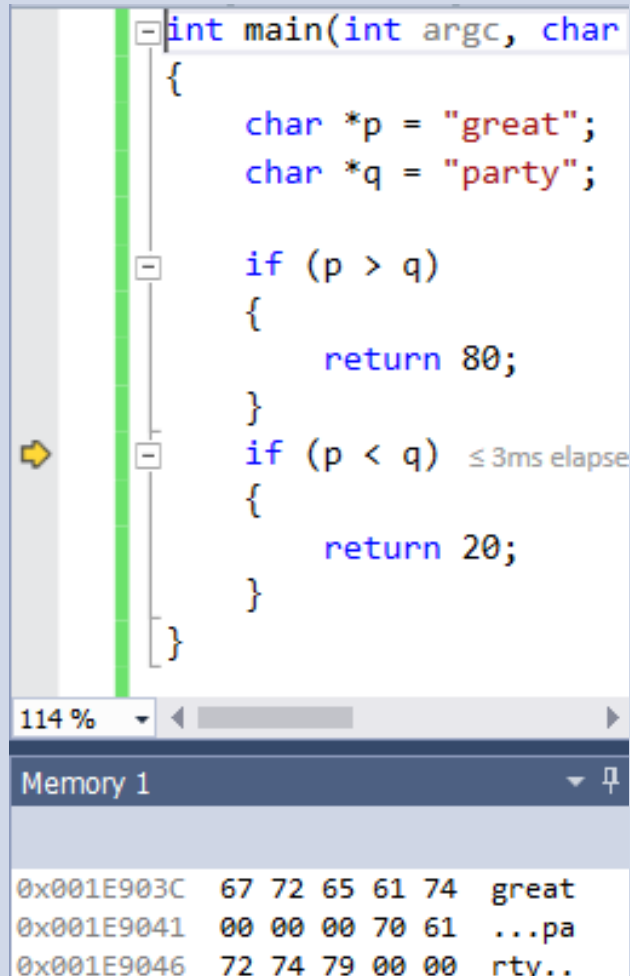
חימום

שגיאות?	קוד
	<pre>int main(int argc, char **argv) { char *p = "great"; char *q = "party"; if (p > q) { return 80; } if (p < q) { return 20; } }</pre>

חימום

שגיאות?

קוד



The screenshot shows a debugger window with a C program. The program defines two character pointers, `p` and `q`, pointing to the strings "great" and "party" respectively. It then compares them using `p > q` and `p < q`. The debugger is currently paused at the `if (p < q)` line, with a yellow arrow pointing to it. The memory dump at the bottom shows the addresses and contents of the strings: `0x001E903C` contains "great", `0x001E9041` contains "...pa", and `0x001E9046` contains "rty..".

```
int main(int argc, char  
{  
    char *p = "great";  
    char *q = "party";  
  
    if (p > q)  
    {  
        return 80;  
    }  
    if (p < q) ≤ 3ms elapse  
    {  
        return 20;  
    }  
}
```

114 %

Memory 1

0x001E903C	67 72 65 61 74	great
0x001E9041	00 00 00 70 61	...pa
0x001E9046	72 74 79 00 00	rty..

```
int main(int argc, char **argv)  
{  
    char *p = "great";  
    char *q = "party";  
  
    if (p > q)  
    {  
        return 80;  
    }  
    if (p < q)  
    {  
        return 20;  
    }  
}
```

חימום

שגיאות?	קוד
	<pre data-bbox="981 428 1812 1119">int main(int argc, char **argv) { char *p = "great"; char *q = "great"; if (p > q) { return 80; } if (p < q) { return 20; } }</pre>

חימום

שגיאות?

קוד

```
int main(int argc, char **argv)
{
    char *p = "great";
    char *q = "great";

    if (p > q)
    {
        return 80;
    }
    if (p < q)
    {
        return 20;
    }
} ≤ 1ms elapsed
```

```
int main(int argc, char **argv)
{
    char *p = "great";
    char *q = "great";

    if (p > q)
    {
        return 80;
    }
    if (p < q)
    {
        return 20;
    }
}
```

האם ה destructors ייקראו או לא?

```
class A { public:
    int a;
    ~A() { printf("destructing A"); }
};

class B { public:
    int a;
    ~B() { printf("destructing B"); }
};

void f(char *input)
{
    A a;
    B b;

    if (strlen(input) < 10)
    {
        throw new int(8); ≤ 4ms elapsed
    }
}

int main(int argc, char **argv)
{
    try
    {
        f("great");
    }
    catch (...)
    {
    }
```

viewing Options

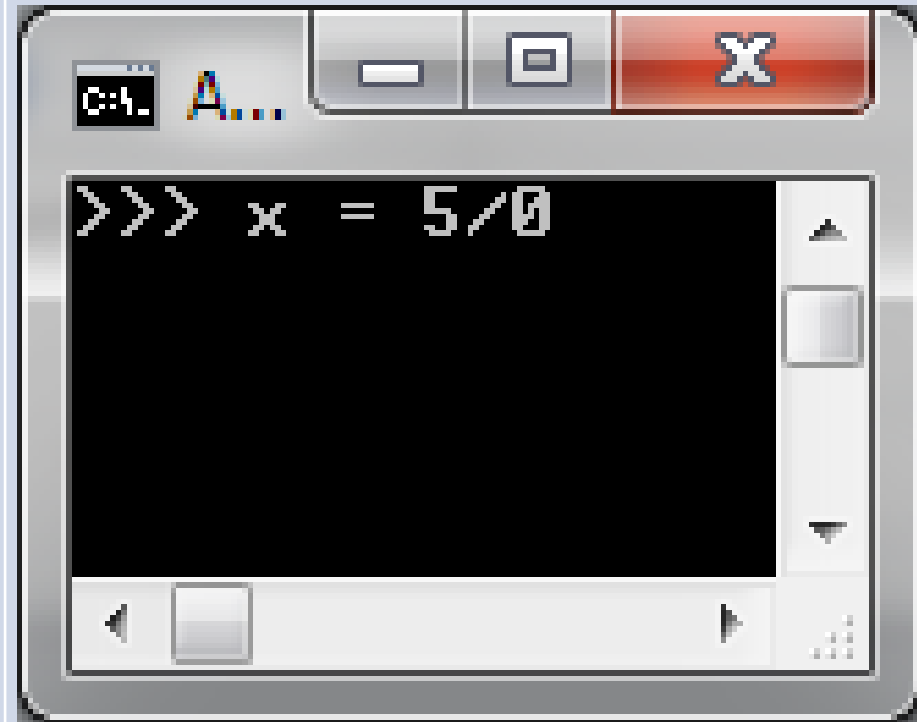
- ☒ Show code bytes
- ☒ Show source code
- ☐ Show line numbers
- ☒ Show address
- ☒ Show symbols

	throw new int(8);		
011A1965	6A 04	push	4
011A1967	E8 5A F9 FF FF	call	operator new (011A12C6h)
011A196C	83 C4 04	add	esp,4
011A196F	89 85 08 FF FF FF	mov	dword ptr [ebp-0F8h],eax
011A1975	83 BD 08 FF FF FF 00	cmp	dword ptr [ebp-0F8h],0
011A197C	74 1A	je	f+88h (011A1998h)
011A197E	8B 85 08 FF FF FF	mov	eax,dword ptr [ebp-0F8h]
011A1984	C7 00 08 00 00 00	mov	dword ptr [eax],8
011A198A	8B 8D 08 FF FF FF	mov	ecx,dword ptr [ebp-0F8h]
011A1990	89 8D 00 FF FF FF	mov	dword ptr [ebp-100h],ecx
011A1996	EB 0A	jmp	f+92h (011A19A2h)
011A1998	C7 85 00 FF FF FF 00 00 00 00	mov	dword ptr [ebp-100h],0
011A19A2	8B 95 00 FF FF FF	mov	edx,dword ptr [ebp-100h]
011A19A8	89 95 14 FF FF FF	mov	dword ptr [ebp-0ECh],edx
011A19AE	68 4C A2 1A 01	push	offset __TI2PAH (011AA24Ch)
011A19B3	8D 85 14 FF FF FF	lea	eax,[ebp-0ECh]
011A19B9	50	push	eax
011A19BA	E8 CF F9 FF FF	call	__CxxThrowException@8 (011A138Eh)
	}		
011A19BF	C6 45 FC 00	mov	byte ptr [ebp-4],0
011A19C3	8D 4D E0	lea	ecx,[b]
011A19C6	E8 E8 F7 FF FF	call	B::~~B (011A11B3h)
011A19CB	C7 45 FC FF FF FF FF	mov	dword ptr [ebp-4],0FFFFFFFFh
011A19D2	8D 4D EC	lea	ecx,[a]
011A19D5	E8 BC F6 FF FF	call	A::~~A (011A1096h)

חימום

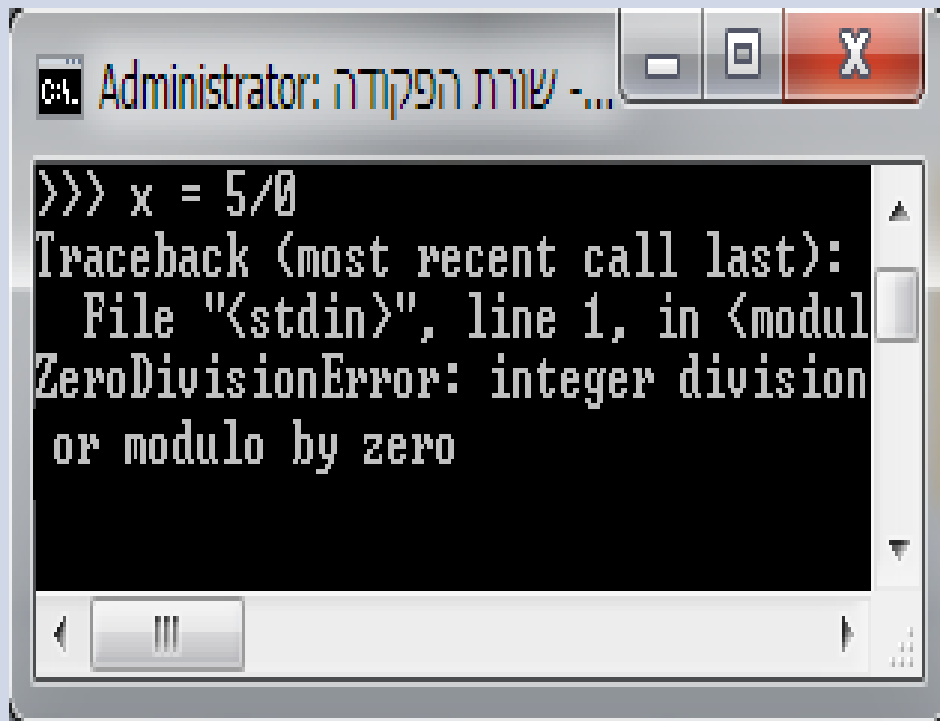
שגיאות?

קוד



חימום

שגיאות?

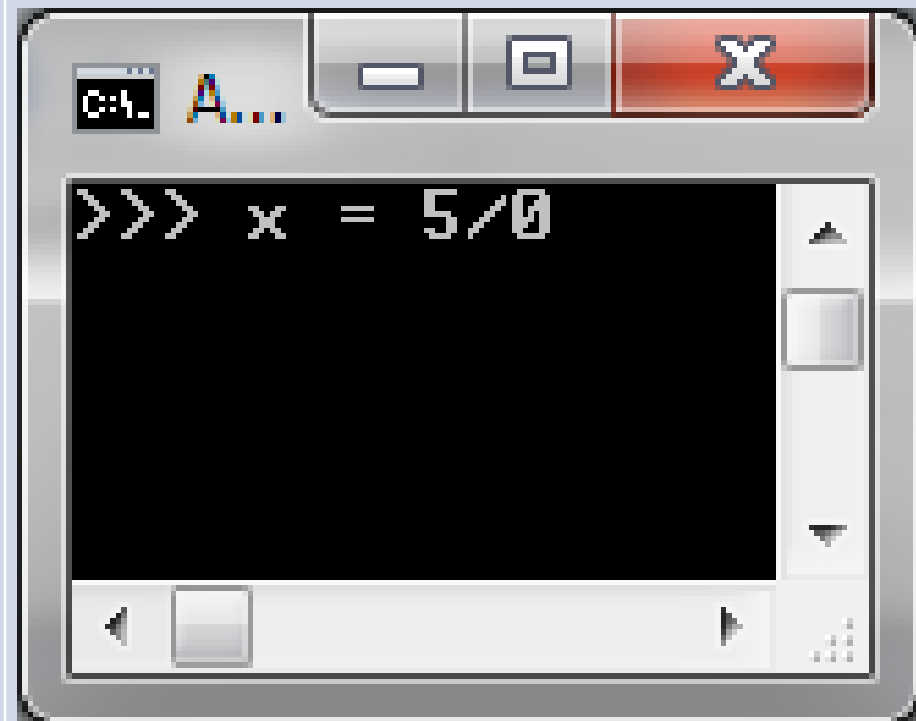


A screenshot of a Windows command prompt window titled "Administrator: שורת הפקודה - ...". The window has standard Windows window controls (minimize, maximize, close). The command prompt shows the following text:

```
>>> x = 5/0  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
ZeroDivisionError: integer division  
or modulo by zero
```

The window has a scrollbar on the right and a status bar at the bottom.

קוד



A screenshot of a Windows command prompt window titled "C:_ A...". The window has standard Windows window controls (minimize, maximize, close). The command prompt shows the following text:

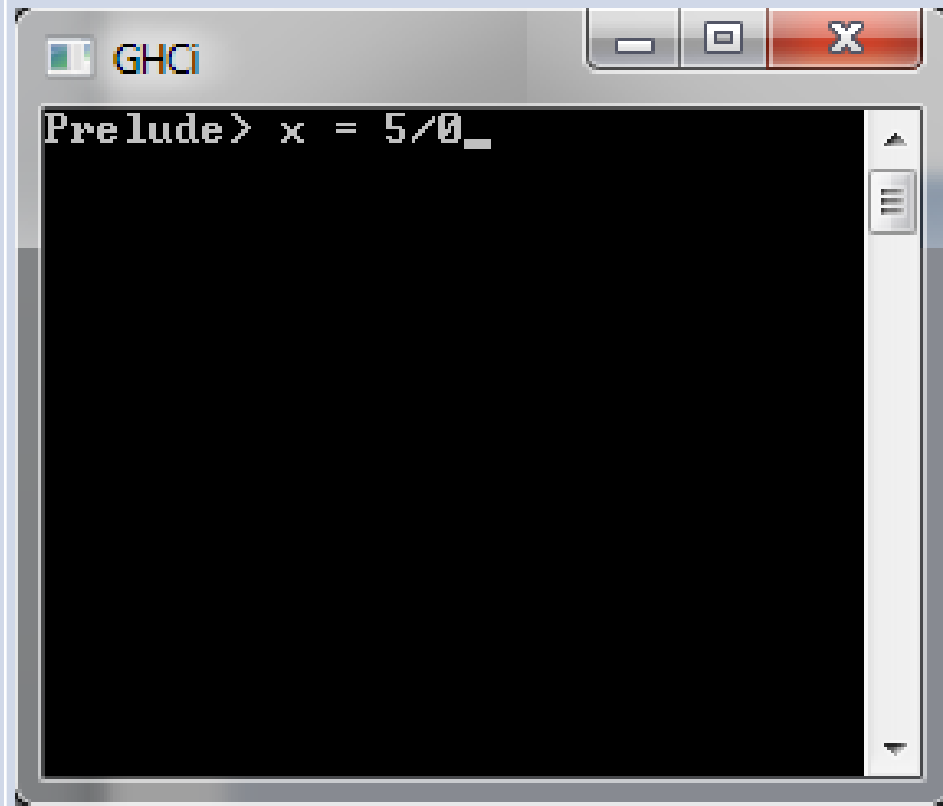
```
>>> x = 5/0
```

The window has a scrollbar on the right and a status bar at the bottom.

חימום

שגיאות?

קוד



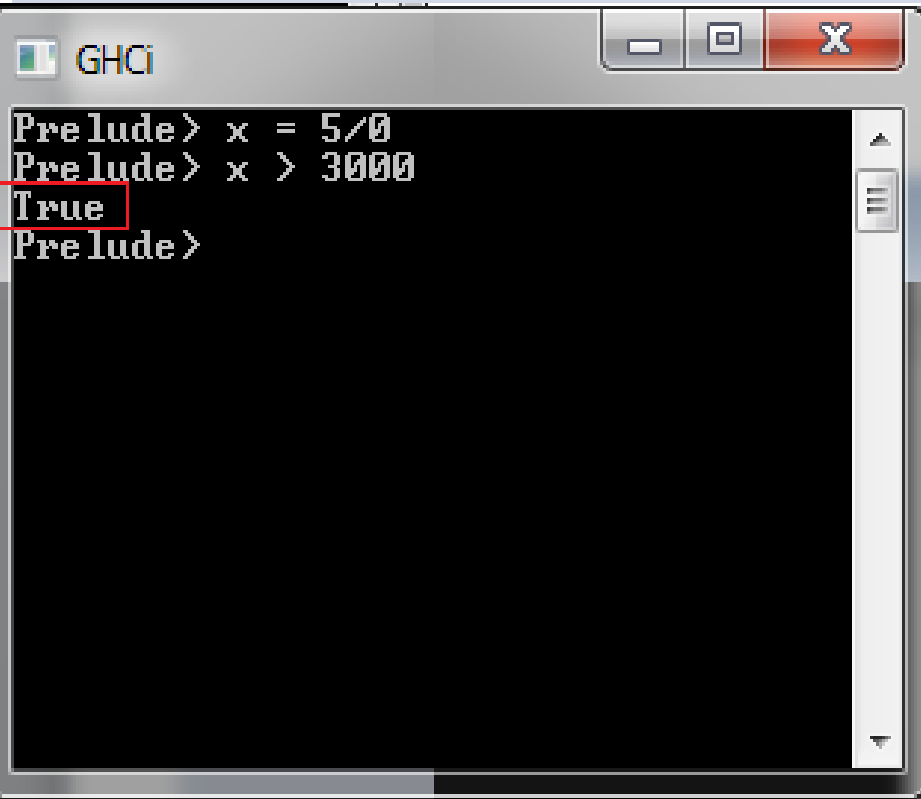
A screenshot of a GHCi terminal window. The window title is "GHCi". The prompt is "Prelude>". The user has entered the expression `x = 5/0_`. The terminal is otherwise empty, with a black background and white text. The window has standard macOS-style window controls (minimize, maximize, close) in the top right corner.

```
GHCi
Prelude> x = 5/0_
```

חימום

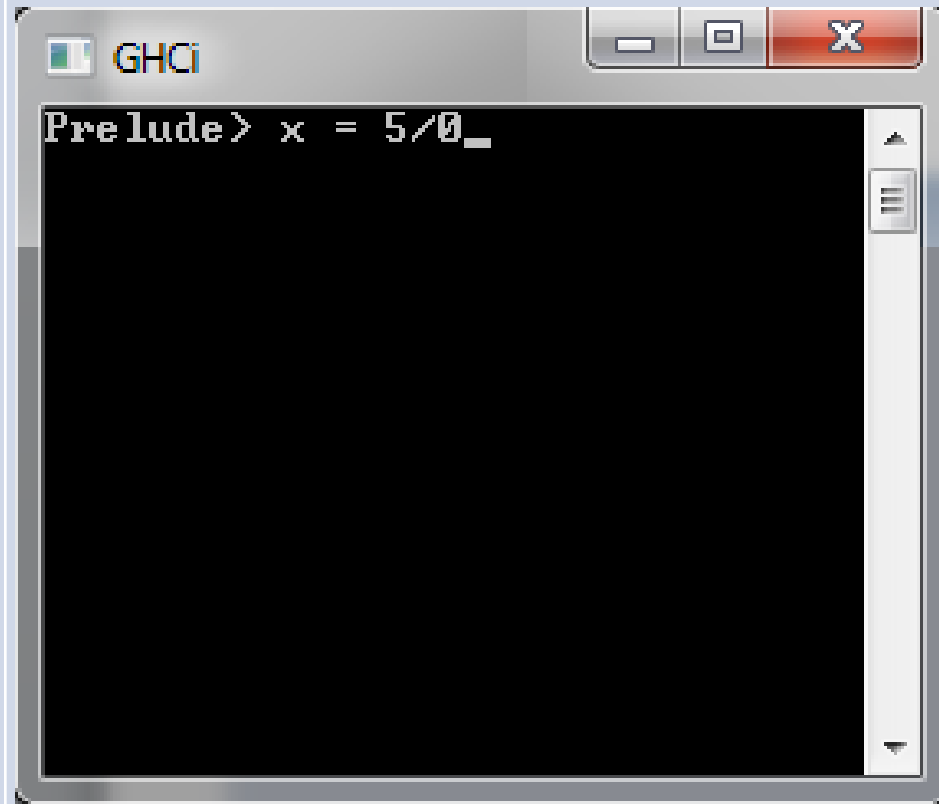
שגיאות?

קוד



A screenshot of a GHCi window. The title bar says 'GHCi'. The prompt is 'Prelude>'. The user has entered 'x = 5/0' and 'x > 3000'. The output is 'True', which is highlighted with a red box. The prompt is now 'Prelude>'.

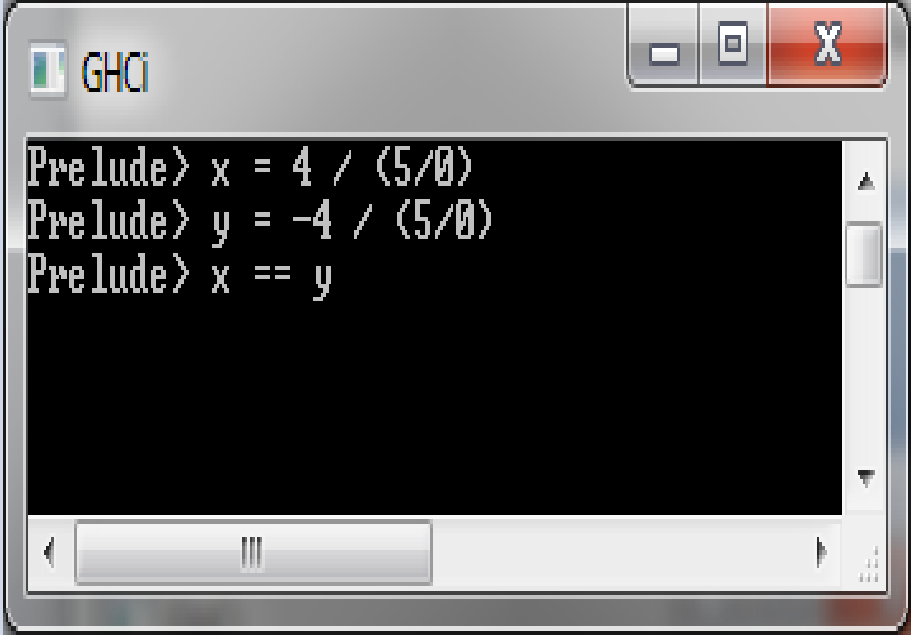
```
Prelude> x = 5/0
Prelude> x > 3000
True
Prelude>
```



A screenshot of a GHCi window. The title bar says 'GHCi'. The prompt is 'Prelude>'. The user has entered 'x = 5/0_'. The output is an empty line. The prompt is now 'Prelude>'.

```
Prelude> x = 5/0_
Prelude>
```

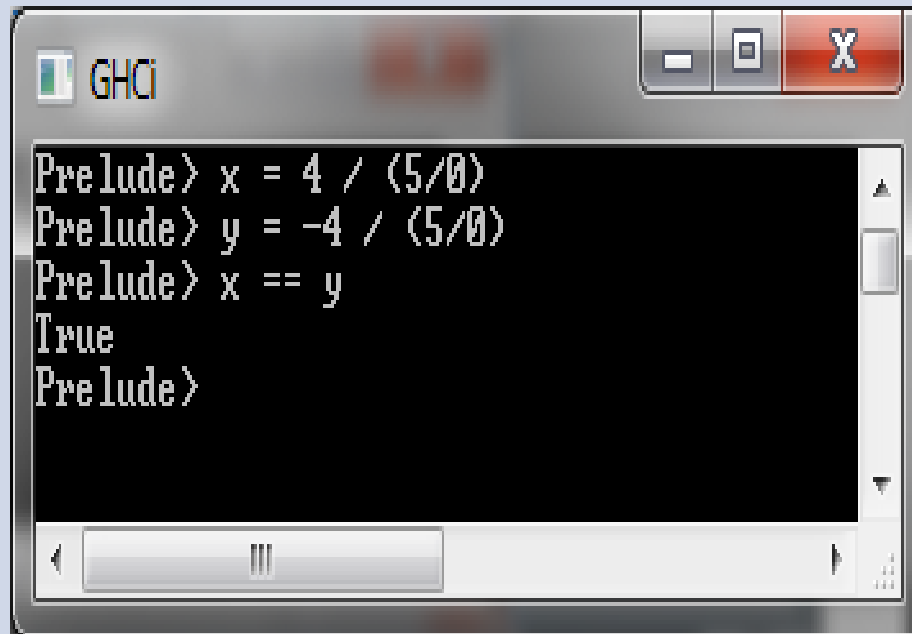
חימום

שגיאות?	קוד
	 <pre>Prelude> x = 4 / (5/0) Prelude> y = -4 / (5/0) Prelude> x == y</pre>

חימום

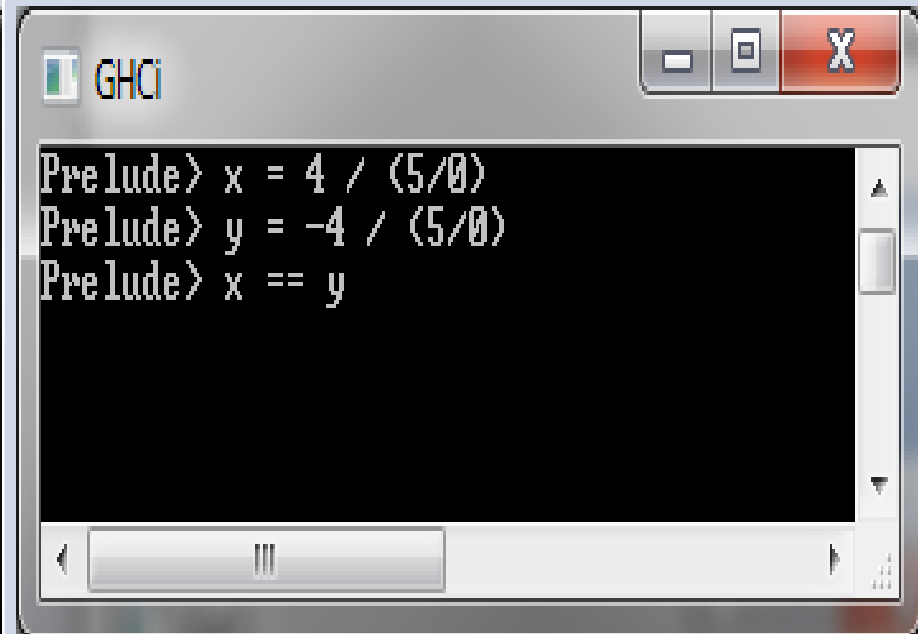
שגיאות?

קוד



A screenshot of a GHCi window. The window has a title bar with the text 'GHCi' and standard window controls (minimize, maximize, close). The main area is a black terminal with white text. The text shows the following sequence of commands and output: 'Prelude> x = 4 / (5/0)', 'Prelude> y = -4 / (5/0)', 'Prelude> x == y', and 'True'. The window has a scrollbar on the right and a status bar at the bottom.

```
Prelude> x = 4 / (5/0)
Prelude> y = -4 / (5/0)
Prelude> x == y
True
Prelude>
```



A screenshot of a GHCi window, identical in appearance to the one on the left. It shows the same sequence of commands: 'Prelude> x = 4 / (5/0)', 'Prelude> y = -4 / (5/0)', and 'Prelude> x == y'. However, the output is missing, and the prompt 'Prelude>' is repeated on the next line, indicating that the program has crashed or is in an error state. The window has a scrollbar on the right and a status bar at the bottom.

```
Prelude> x = 4 / (5/0)
Prelude> y = -4 / (5/0)
Prelude> x == y
Prelude>
```

Semantic Analysis

- מה אפשר להגיד על המשפט הבא: הים התיכון קיבל 96 בבגרות באנגלית. האם הוא נכון לקסיקלית? תחבירית?
- ניתוח סמנטי הוא שלב בו נבדקת משמעות המשפט.
- נתחיל מהדוגמא של שפת המחשבון. עד עתה היינו מעוניינים לדעת האם הביטוי החשבוני שמולנו הוא חוקי. למשל, $50-6 + 34$ אינו חוקי.
- עכשיו, בהינתן ביטוי חוקי, נרצה לחשב את הערך שלו

– חישוב הערך של ביטוי חשבוני חוקי –
באמצעות top down parser

- הנה הדקדוק של שפת המחשבון עם קדימויות אופרטורים:

$$F \rightarrow \text{INT}$$

$$T \rightarrow T * F$$

$$E \rightarrow E + T$$

$$F \rightarrow (E)$$

$$T \rightarrow T / F$$

$$E \rightarrow E - T$$

$$T \rightarrow F$$

$$E \rightarrow T$$

- האם אפשר לבנות לו top down parser?

חישוב הערך של ביטוי חשבוני חוקי –

באמצעות top down parser

אחרי ביטול רקורסיה שמאלית:

$$F \rightarrow \text{INT}$$

$$F \rightarrow (E)$$

$$T \rightarrow FT'$$

$$T' \rightarrow * FT'$$

$$T' \rightarrow / FT'$$

$$T' \rightarrow \epsilon$$

$$E \rightarrow TE'$$

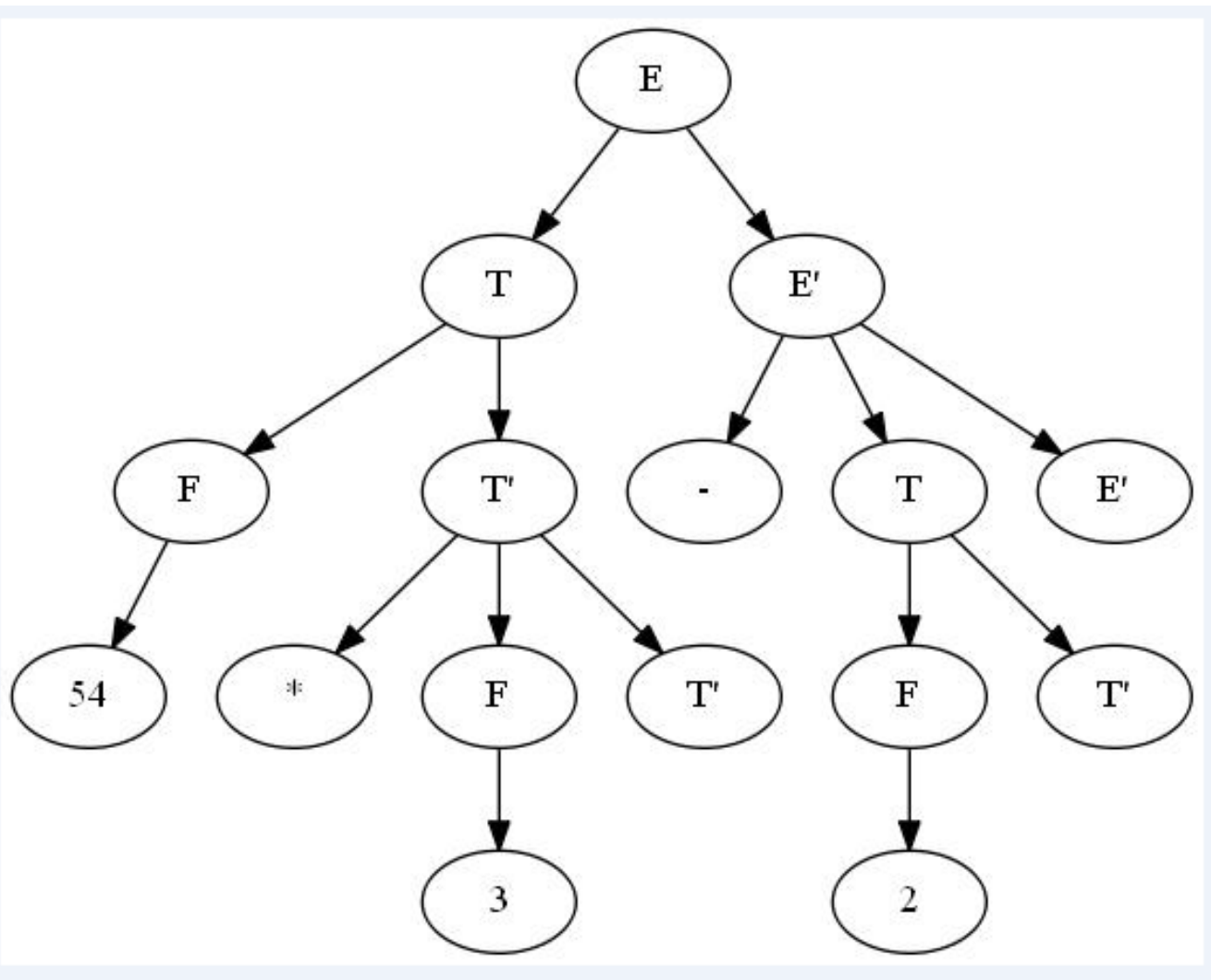
$$E' \rightarrow + TE'$$

$$E' \rightarrow - TE'$$

$$E' \rightarrow \epsilon$$

- איך נשתלב בקוד הקיים לחישוב ערך הביטוי?
- במקור הייתה פונקציה לכל משתנה, וניתן היה להסיק בקלות את סדר כללי הגזירה שהופעל כדי להגיע למילה. אבל, עץ הגזירה לא נבנה במפורש.

עץ הגזירה של הביטוי $54*3-2$




```

T_tag_type T_tag()
{
    T_tag_type t_tag = (T_tag_type) malloc(sizeof(*t_tag));

    switch (tok) {

    case (TIMES):

        // T' ---> * F T'
        Eat(TIMES);
        t_tag->op = TIMES;
        t_tag->F = F();
        t_tag->T_tag = T_tag();
        return t_tag;

    case (DIVIDE):

        // T' ---> / F T'
        Eat(DIVIDE);
        t_tag->op = DIVIDE;
        t_tag->F = F();
        t_tag->T_tag = T_tag();
        return t_tag;
    }

    free(t_tag);
    // T' ---> epsilon
    return NULL;
}

```

חישוב הערך של ביטוי חשבוני חוקי – באמצעות top down parser

- במעבר ראשון על הקלט נבנה את עץ הגזירה.
- לכל משתנה מתאים struct שיכיל את הנגזרים:
- המשתנה F יוצא דופן כאן – למה?

```
struct E_type_ {  
    T_type T;  
    E_tag_type E_tag;  
};
```

```
struct T_type_ {  
    F_type F;  
    T_tag_type T_tag;  
};
```

```
struct F_type_ {  
    int value;  
    E_type E;  
};
```

```
struct E_tag_type_ {  
    int op;  
    T_type T;  
    E_tag_type E_tag;  
};
```

```
struct T_tag_type_ {  
    int op;  
    F_type F;  
    T_tag_type T_tag;  
};
```

חישוב הערך של ביטוי חשבוני חוקי –

באמצעות top down parser

- חישוב הערך מתבצע על ידי מעבר על עץ הגזירה. שמאל לפני ימין. קדימויות האופרטורים כבר מובנות בעץ.
- בשקף הבא הקוד שמשערך רקורסיבית את עץ הגזירה. מימוש סטנדרטי.

```

int Eval_F(F_type f)
{
    if (f->E == NULL) return f->value;
    else return Eval_E(f->E);
}

int Eval_T(T_type t)
{
    if (t->T_tag == NULL) return Eval_F(t->F);
    else if (t->T_tag->op == TIMES) return Eval_F(t->F) * Eval_T_tag(t->T_tag);
    else if (t->T_tag->op == DIVIDE) return Eval_F(t->F) / Eval_T_tag(t->T_tag);
}

int Eval_T_tag(T_tag_type t_tag)
{
    if (t_tag->T_tag == NULL) return Eval_F(t_tag->F);
    else if (t_tag->T_tag->op == TIMES) return Eval_F(t_tag->F) * Eval_T_tag(t_tag->T_tag);
    else if (t_tag->T_tag->op == DIVIDE) return Eval_F(t_tag->F) / Eval_T_tag(t_tag->T_tag);
}

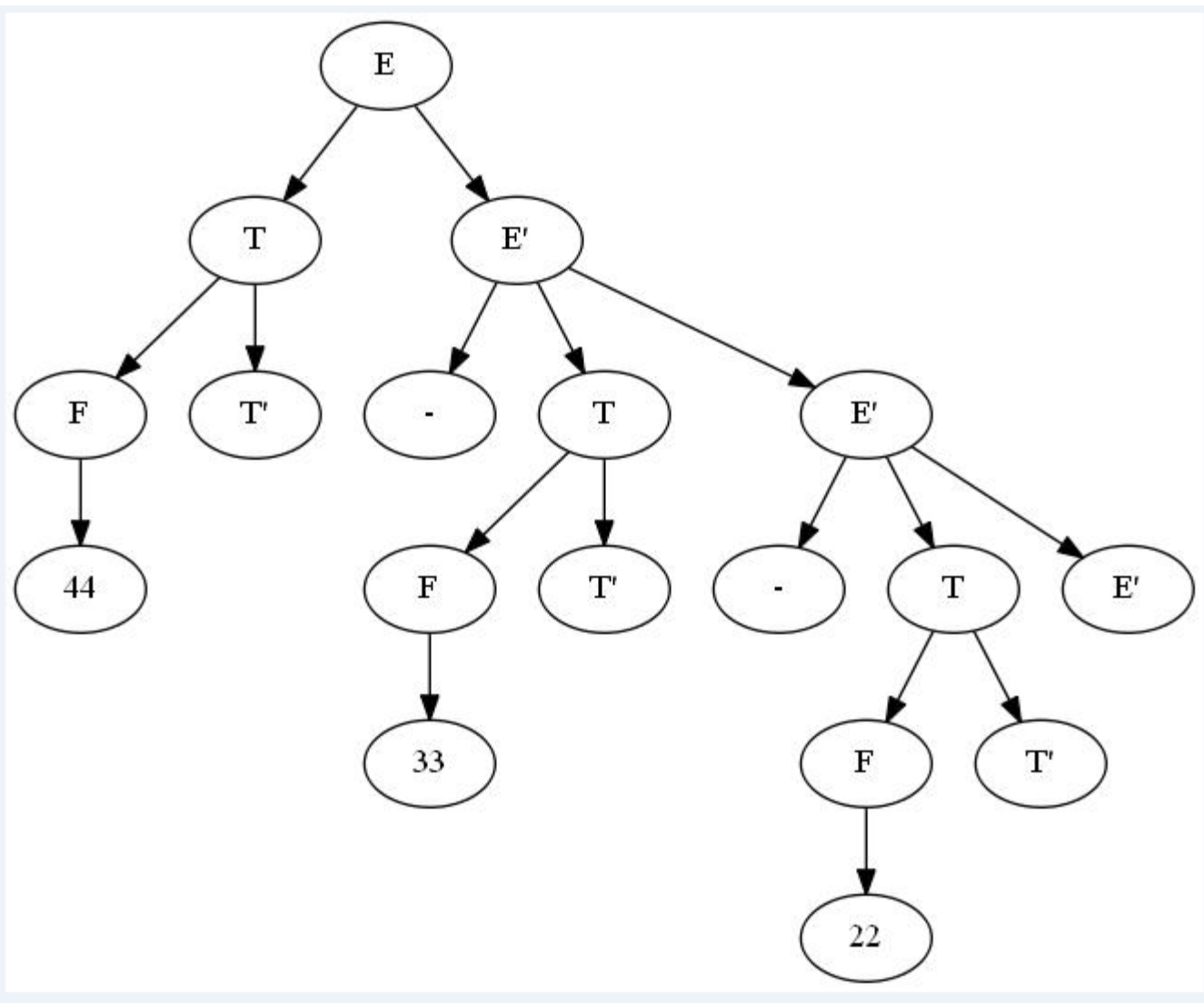
int Eval_E_tag(E_tag_type e_tag)
{
    if (e_tag->E_tag == NULL) return Eval_T(e_tag->T);
    else if (e_tag->E_tag->op == PLUS) return Eval_T(e_tag->T) + Eval_E_tag(e_tag->E_tag);
    else if (e_tag->E_tag->op == MINUS) return Eval_T(e_tag->T) - Eval_E_tag(e_tag->E_tag);
}

int Eval_E(E_type e)
{
    if (e->E_tag == NULL) return Eval_T(e->T);
    else if (e->E_tag->op == PLUS) return Eval_T(e->T) + Eval_E_tag(e->E_tag);
    else if (e->E_tag->op == MINUS) return Eval_T(e->T) - Eval_E_tag(e->E_tag);
}

```

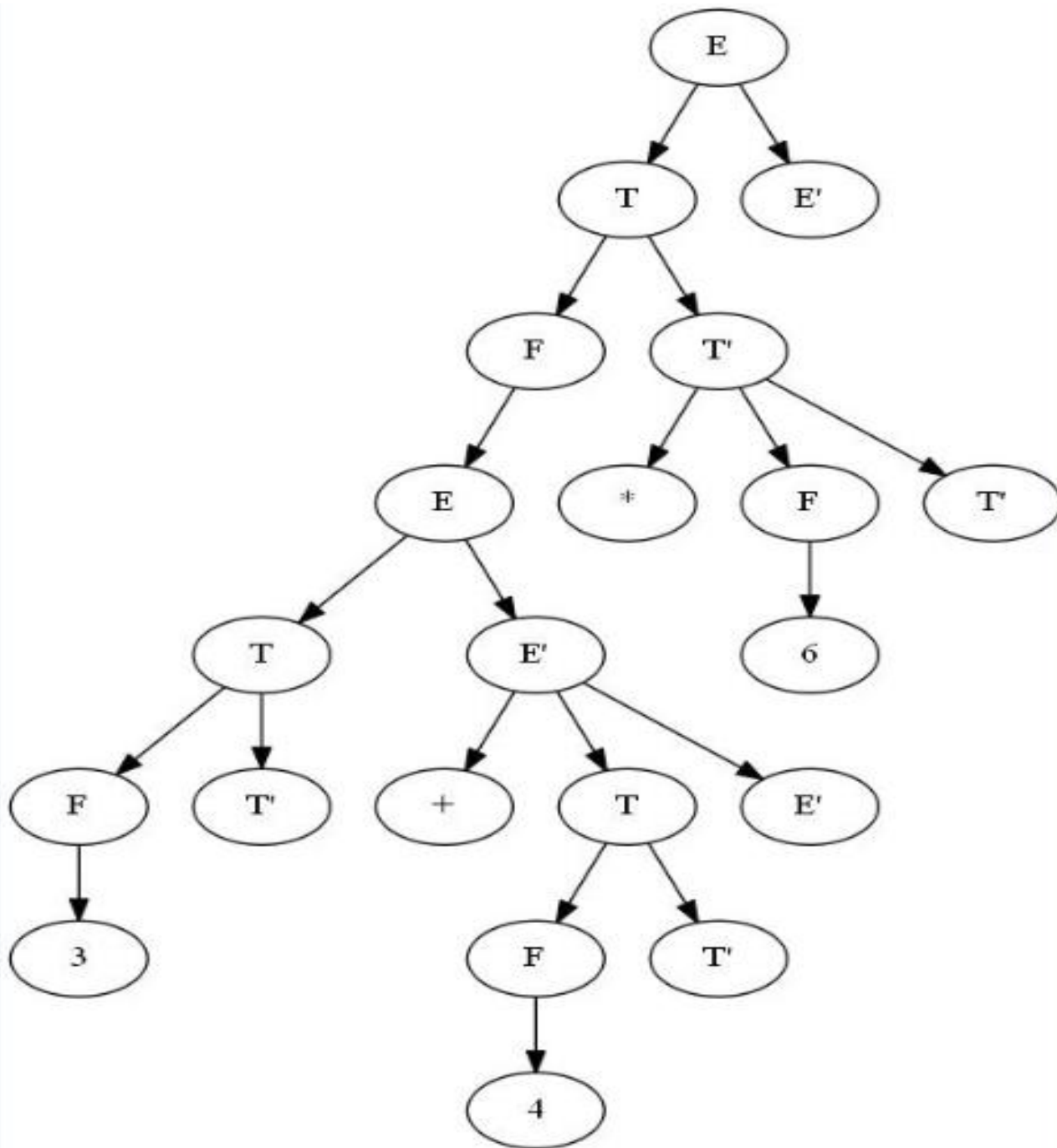
עוד נקודות בקשר לעץ הגזירה

- האם הפעולות חיסור וחילוק אסוציאטיביות לשמאל או לימין?
- האם המידע הזה קיים בעץ הגזירה?
- האם האסוציאטיביות היא כמו של הדקדוק המקורי לפני ביטול הרקורסיה השמאלית?
- הציור הבא יוכל לתת לנו את התשובה..



מי משגנע את עץ הגזירה שלי?

- בשקף הבא נראה עץ גזירה שנוצר אוטומטית בדיוק באותה צורה כמו העצים הקודמים. אבל, משום מה, יש בו קדימות של חיבור על כפל.
- הסתכלו היטב וגלו מה קרה שם!



תשובה..

- הביטוי המקורי הוא $6 * (3+4)$ אבל את הסוגריים לא רואים ב AST זהו בעצם אחד ההבדלים בין עץ גזירה ל AST הורדת סימנים מיותרים. באופן דומה, ביטוי של גישה למערך $A[i+4]$ ישמור את שם המערך, ואינדקס הגישה, ללא הסוגריים המרובעים.
- ההבדל השני, והחשוב יותר, הוא שעבור טרמינלים עם ערך (כמו INT למשל) הערך נמצא גם כן ב AST ובאופן פורמלי, לא נמצא בעץ הגזירה.

חישוב הערך של ביטוי חשבוני חוקי –

באמצעות LR(1) parser (bison)

- המשתנה program הוא המשתנה הראשון (שמסומן בדקדוקים S)
- כאשר מתבצע reduce לכלל כלשהו, אז הקוד בתוך הסוגריים מימין לכלל מתבצע. המשמעות של \$\$ היא תת העץ שמייצג את הביטוי. המשמעות של \$1, \$2, \$3 וכו' היא תתי העצים (או טוקנים) בתוך הכלל עצמו, ממוספרים משמאל לימין

```
%start program
```

```
%%
```

```
program:      E                {printf("%d\n", $$ . gval . ival);}
```

```
E:           INT                { $$ . gval . ival = $1 . gval . ival; }  
  | E PLUS E  { $$ . gval . ival = $1 . gval . ival + $3 . gval . ival; }  
  | E MINUS E  { $$ . gval . ival = $1 . gval . ival - $3 . gval . ival; }  
  | E TIMES E  { $$ . gval . ival = $1 . gval . ival * $3 . gval . ival; }  
  | E DIVIDE E { $$ . gval . ival = $1 . gval . ival / $3 . gval . ival; }  
  | LPAREN E RPAREN { $$ . gval . ival = $2 . gval . ival; }
```

```
%%
```

הרחבת שפת המחשבון כך שתכיל גם floats – גישה 1

- איך נוכל להרחיב את הדקדוק כך שהמחשבון שלנו ידע לעשות פעולות גם על floats
- מה אתם אומרים על האפשרות של להוסיף כללי גזירה $S \rightarrow \text{FLOAT}$ וזהו? זה מספיק?
- מה משמעות הביטוי $5 + 4.44$?
- מה לגבי $1.0 / 4 / 2$?

```

program:      E      {
                  if ($$.type == 0) {printf("%d\n",$$$.gval.ival);}
                  if ($$.type == 1) {printf("%f\n",$$$.gval.fval);}
                  }

E:            INT     {$$$.gval.ival = $1.gval.ival; $$$.type = 0;}
              | FLOAT {$$$.gval.fval = $1.gval.fval; $$$.type = 1;}
              | E DIVIDE E {
                  if (($1.type == 0) && ($3.type == 0))
                  {
                      $$$.type=0;
                      $$$.gval.ival = $1.gval.ival / $3.gval.ival;
                  }
                  if (($1.type == 1) && ($3.type == 1))
                  {
                      $$$.type=1;
                      $$$.gval.fval = $1.gval.fval / $3.gval.fval;
                  }
                  if (($1.type == 0) && ($3.type == 1))
                  {
                      $$$.type=1;
                      $$$.gval.fval = $1.gval.ival / $3.gval.fval;
                  }
                  if (($1.type == 1) && ($3.type == 0))
                  {
                      $$$.type=1;
                      $$$.gval.fval = $1.gval.fval / $3.gval.ival;
                  }
                  }

              | E PLUS E {$$$.gval.ival = $1.gval.ival + $3.gval.ival;}
              | E MINUS E {$$$.gval.ival = $1.gval.ival - $3.gval.ival;}

```

הרחבת שפת המחשבון כך שתכיל גם floats – גישה 2

```
program:      I      {printf("%d\n",$$.$val.ival);}
              | F      {printf("%f\n",$$.$val.fval);}

I:            INT      {$$.$val.ival = $1.$val.ival;}
              | I PLUS I {$$.$val.fval = $1.$val.ival + $3.$val.ival;}
              | I DIVIDE I {$$.$val.ival = $1.$val.ival / $3.$val.ival;}
              | I MINUS I {$$.$val.ival = $1.$val.ival - $3.$val.ival;}
              | I TIMES I {$$.$val.ival = $1.$val.ival * $3.$val.ival;}
              | LPAREN I RPAREN {$$.$val.ival = $2.$val.ival;}

F:            FLOAT      {$$.$val.fval = $1.$val.fval;}
              | F PLUS F {$$.$val.fval = $1.$val.fval + $3.$val.fval;}
              | F DIVIDE F {$$.$val.fval = $1.$val.fval / $3.$val.fval;}
              | F MINUS F {$$.$val.fval = $1.$val.fval - $3.$val.fval;}
              | F TIMES F {$$.$val.fval = $1.$val.fval * $3.$val.fval;}
              | F PLUS I {$$.$val.fval = $1.$val.fval + $3.$val.ival;}
              | F DIVIDE I {$$.$val.fval = $1.$val.fval / $3.$val.ival;}
              | F MINUS I {$$.$val.fval = $1.$val.fval - $3.$val.ival;}
              | F TIMES I {$$.$val.fval = $1.$val.fval * $3.$val.ival;}
              | I PLUS F {$$.$val.fval = $1.$val.ival + $3.$val.fval;}
              | I DIVIDE F {$$.$val.fval = $1.$val.ival / $3.$val.fval;}
              | I MINUS F {$$.$val.fval = $1.$val.ival - $3.$val.fval;}
              | I TIMES F {$$.$val.fval = $1.$val.ival * $3.$val.fval;}
              | LPAREN F RPAREN {$$.$val.fval = $2.$val.fval;}
```

Abstract Syntax Tree (AST)

- כדי לשמר מודלריות, נרצה להפריד את שלב ה syntax analysis משלב ה semantic analysis.
- להפריד בין בדיקת תקינות מבנה המשפט לבין הבנת משמעות המשפט.
- לצורך כך, ה bison יבנה abstract syntax tree, שיועבר לשלבים הבאים של הקומפיילר.
- ה AST הוא ייצוג מלא של התוכנית המקורית. מרגע שנוצר ה AST, קובץ התוכנית המקורי אינו נחוץ כלל, ותפיסתית ניתן למחוק אותו.
- כדי לתת דוגמאות ל AST, צריך להציג כבר עכשיו את שפר המקור שלנו: StarKist

Abstract Syntax Tree

- אנחנו נפרדים כעת משפת המחשבון ועוברים לשפת StarKist שתלווה אותנו עד סוף הקורס.
הפרוייקט העיקרי שלנו יהיה לבנות קומפיילר עבורה מתחילתו ועד סופו.
- שפת StarKist היא שפה מונחת עצמים מומצאת, שמכילה אלמנטים מתקדמים כמו בדיקות זמן ריצה נגד חריגות גישה, משתנים שמוגדרים כ auto ועוד.
- בשקפים הבאים נתאר את StarKist

שפת StarKist שלנו: טיפוסים השפה

- בשפה יש שני טיפוסים בסיסיים: `string` ו `int`
- הגדרת משתנה נעשית באופן הבא: `var i:int`
כלומר, `i` הוא משתנה מסוג `int`
- מותר להגדיר משתנים ללא טיפוס, כמו ב `C++`
שמרשה הגדרת `auto`, והאתחול קובע את הטיפוס:
- `var i := 8+15*9` הוא משתנה מסוג `int`
- `var s := "dudu"` הוא משתנה מסוג `string`

שפת StarKist שלנו: פעולות בינאריות

- בדומה למרבית שפות התכנות הקיימות, אופרטורים בינאריים מוגדרים רק על הטיפוסים הפרימיטיביים `int` ו `string`.
- עבור `string` מותרות הפעולות הבאות: `=` השוואה לקסיקוגרפית, `+` שרשור בין מחרוזות.
- עבור `int`-ים מותרות הפעולות `<`, `>`, `=`, `+`, `-`, `*`, `/`

שפת StarKist שלנו: מבני בקרה (if, while, for)

- בדומה לשפת C:
- `var i:int := 8`
- `If (i > 6){ i := i+4 }`
- בתוך הסוגריים של ה `if` מופיע ערך מסוג `int`
- אם הערך שונה מ `0`, ה `if` נלקח. אחרת לא.
- לפעולה `i > 6` יש ערך מספרי מטיפוס `int` והוא שווה לאחד אם זה פסוק אמת או `0` אחרת.

שפת tiger שלנו: דוגמא פשוטה

- תוכנית היא ביטוי `let` אשר בחלקו הראשון (**אדום**) מופיעות הגדרות משתנים, מבנים ופונקציות, ובחלקו השני (**כחול**) באות הפקודות לביצוע. לביטוי `let` יש ערך מוחזר והוא ערכו של הביטוי האחרון בסדרה שבוצעה.
- אצלנו בדוגמא, יוחזר `int`

`let`

```
var i:int := 1  
var j:int := 80
```

`in`

```
i+j
```

`end`

שפת tiger שלנו: דוגמא נוספת

let

```
function IsPrime(p:int) : int =  
  let  
    var returnedValue := 1  
  in  
    (  
      for i := 2 to p-1 do  
        for j := 2 to p-1 do  
          if (i*j = p) then  
            returnedValue := 0;  
          returnedValue  
        )  
      end
```

in

```
  for i := start to finish do  
    if IsPrime(i) then  
      PrintInt(i)
```

end

הריבוע הירוק, הוא מימוש
הפונקציה IsPrime, שלה יש
משתנה לוקאלי אחד:
returnedValue.

הפונקציה מבצעת סדרה
של שתי פעולות (מופרדות
בנקודה-פסיק, ועטופות
בסוגריים). הערך המוחזר
הוא ערכו של הביטוי
האחרון בסדרה, במקרה זה
השני: ערך המשתנה
returnedValue.

איפה יכולים להיות מוגדרים
המשתנים finish ו start
מהמשך?

שפת tiger שלנו: פונקציות מקוננות

```
let
function PrintPrimesInRange(start:int,finish:int) =
let
function IsPrime(p:int) : int =
...
in
for i := start to finish do
if IsPrime(i) then
PrintInt(i)
end
in
PrintPrimesInRange(2,100)
end
```

אפשר לראות כאן
פונקציות מקוננות:
הפונקציה **IsPrime**
נגישה רק מתוך הגוף
של הפונקציה
PrintPrimesInRange

(הקוד של IsPrime
הושמט כאן משיקולי
מקום, ונמצא בשקף
הקודם..)

האם פונקציות
מקוננות קיימות
בשפת C? מה כן?

בניית הדקדוק לשפת tiger

```
%start program
```

```
%%
```

```
program:      exp                {absyn_root = $1.gval.exp;}

exp:          INT                {$$.$gval.exp = A_IntExp(EM_tokPos,$1.gval.ival);}
           | FLOAT              {$$.$gval.exp = A_FloatExp(EM_tokPos,$1.gval.fval);}
           | STRING             {$$.$gval.exp = A_StringExp(EM_tokPos,$1.gval.sval);}
           | lvalue             {$$.$gval.exp = A_VarExp(EM_tokPos,$1.gval.var);}
           | SequenceExp        {$$.$gval.exp = $1.gval.exp;}
           | LPAREN exp RPAREN  {$$.$gval.exp = $2.gval.exp;}
           | LetExp              {$$.$gval.exp = $1.gval.exp;}
           | ForExp              {$$.$gval.exp = $1.gval.exp;}
           | IfThenExp           {$$.$gval.exp = $1.gval.exp;}
           | AssignExp           {$$.$gval.exp = $1.gval.exp;}
           | OpExp               {$$.$gval.exp = $1.gval.exp;}
           | CallExp             {$$.$gval.exp = $1.gval.exp;}
```

- עץ הגזירה מתחיל ב program.
- A_FloatExp, A_IntExp וכו' מחזירים אותו טיפוס, מצביע ל A_Exp_ שהוא union ענקי.
- המשתנה absyn_root הוא גם מטיפוס מצביע ל A_Exp_ והוא מוגדר בקובץ C
- האפשרויות השונות מתארות את סוגי הביטויים הקיימים
- בשקף הבא נתמקד ב LetExp

TypeFields:	ID COLON ID COMMA TypeFields ID COLON ID	{\$.gval.fieldList = A_FieldList(A_Field(...),\$.gval.fieldList);} {\$.gval.fieldList = A_FieldList(A_Field(...),NULL);}
TypeDeclaration:	TYPE ID EQ ID TYPE ID EQ ARRAY OF ID TYPE ID EQ LBRACE TypeFields RBRACE	{\$.gval.dec = A_TypeDec(EM_tokPos,A_Namety(S_Symbol(...),A_NameTy(...))));} {\$.gval.dec = A_TypeDec(EM_tokPos,A_Namety(S_Symbol(...),A_ArrayTy(...))));} {\$.gval.dec = A_TypeDec(EM_tokPos,A_Namety(S_Symbol(...),A_RecordTy(...))));}
declaration:	TypeDeclaration VariableDeclaration FunctionDeclaration	{\$.gval.dec = \$.gval.dec;} {\$.gval.dec = \$.gval.dec;} {\$.gval.dec = \$.gval.dec;}
declarations:	declaration declarations declaration	{\$.gval.decList = A_DeclList(\$.gval.dec,\$2.gval.decList);} {\$.gval.decList = A_DeclList(\$.gval.dec,NULL);}
LetExp:	LET declarations IN exp END	{\$.gval.exp = A_LetExp(EM_tokPos,\$2.gval.decList,\$4.gval.exp);}

Abstract Syntax Tree: C File Implementation

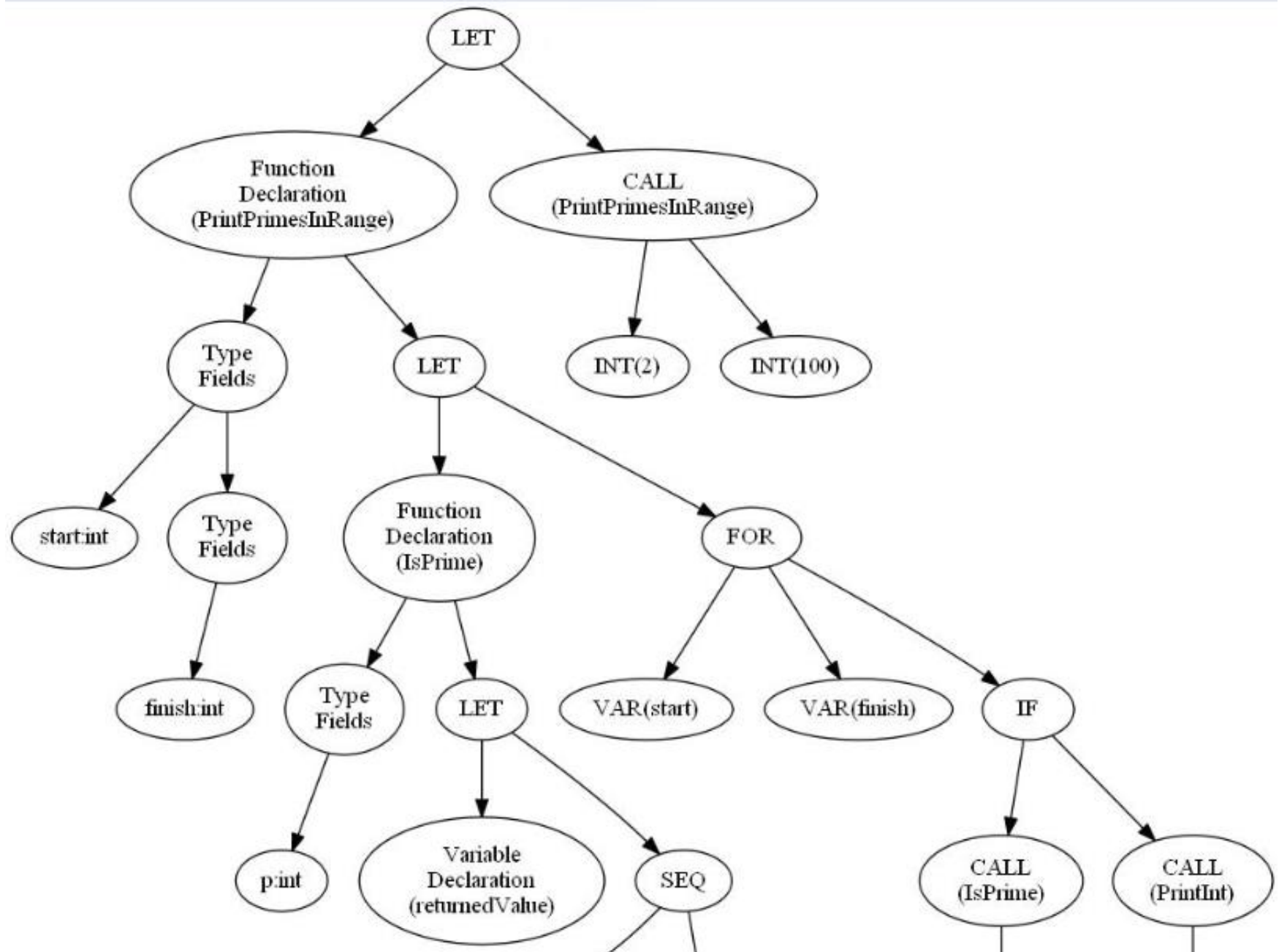
```
struct A_exp_  
{  
    expType kind;  
  
    A_pos pos;  
  
    union  
    {  
        A_var var;  
        /* nil; - needs only the pos */  
        int intt;  
        float floatt;  
        string stringg;  
        struct {S_symbol func; A_expList args; int numArgs;} call;  
        struct {A_oper oper; A_exp left; A_exp right;} op;  
        struct {S_symbol typ; A_efieldList fields;} record;  
        A_expList seq;  
        struct {A_var var; A_exp exp;} assign;  
        struct {A_exp test, then, elsee;} iff; /* elsee is optional */  
        struct {A_exp test, body;} whilee;  
        struct {S_symbol var; A_exp lo,hi,body; bool escape;} forr;  
        /* breakk; - need only the pos */  
        struct {A_declList decs; A_exp body;} let;  
        struct {S_symbol typ; A_exp size, init;} array;  
        struct {S_symbol reocrdType; A_expList initExpList;} recordInit;  
        struct {S_symbol arrayType; int size;} arrayInit;  
    } u;  
};
```


LetExp – Constructor

```
A_exp A_LetExp(A_pos pos, A_decList decs, A_exp body)
{
    A_exp p = (A_exp) checked_malloc(sizeof(*p));
    p->kind=A_letExp;
    p->pos=pos;
    p->u.let.decs=decs;
    p->u.let.body=body;
    return p;
}
```

- שימו לב לבנאי של ביטוי let – רק למלא את השדות במקום המתאים. שום דבר מעבר. באותו אופן פועלים כל הבנאים של שאר הביטויים

Abstract Syntax Tree



Abstract Syntax Tree

- למה לשמור ב AST את המיקומים שה Lexer מוציא? ובכלל, על איזה מיקום בדיוק מדובר?

```
declarations: declaration declarations    {$$.gval.decList = A_DecList($1.gval.dec,$2.gval.decList);}
              | declaration              {$$.gval.decList = A_DecList($1.gval.dec,NULL);}
              ;

LetExp:      LET declarations IN exp END  {$$.gval.exp = A_LetExp(EM_tokPos,$2.gval.decList,$4.gval.exp);}
              ;

lvalue:      ID                          {$$.gval.var = A_SimpleVar(EM_tokPos,S_Symbol($1.gval.sval));}
              | lvalue ARROW ID           {$$.gval.var = A_FieldVar(EM_tokPos,$1.gval.var,S_Symbol($3.gval.sval));}
              | lvalue LBRACK exp RBRACK  {$$.gval.var = A_SubscriptVar(EM_tokPos,$1.gval.var,$3.gval.exp);}

AssignExp:   lvalue ASSIGN exp            {$$.gval.exp = A_AssignExp(EM_tokPos,$1.gval.var,$3.gval.exp);}
              | lvalue ASSIGN AllocateArrayExp {$$.gval.exp = A_AssignExp(EM_tokPos,$1.gval.var,$3.gval.exp);}
              | lvalue ASSIGN AllocateRecordExp {$$.gval.exp = A_AssignExp(EM_tokPos,$1.gval.var,$3.gval.exp);}
```

Semantic Analysis

- מהרגע שיש בידנו AST, אנחנו בשלים לעבור לשלב הבא – Semantic Analysis
- בשלב זה נבדוק האם התוכנית התקינה מבחינת syntax היא גם כן תקינה מבחינה סמנטית.
- נבדוק, למשל, את הדברים הבאים:
- האם משתמשים במשתנה לפני שהוגדר?
- האם יש גישה ב struct לשדה שאינו קיים?
- פונקציה נקראת עם מספר פרמטרים לא נכון?
- יש השמה בין טיפוסים לא מתאימים?

Semantic Analysis

- האלגוריתם של semantic analysis רץ על ה AST, ובאופן לא מפתיע הוא רקורסיבי. בכל שלב הוא יבדוק את העץ שלפניו לפי סוגו (קריאה לפונקציה, שליפת שדה מ struct, השמה וכו') וימשיך לבדוק את תתי העצים שלו ברקורסיה.
- הודעות השגיאה בשלב זה הן רבות ומגוונות יותר. כמו כן, בהרבה מקרים, הקומפיילר מסוגל להגדיר בדיוק רב את הטעות שקרתה

Semantic Analysis – scanning the AST recursively for errors

```
= /*****/  
/* SEM_transExp */  
/*****/  
= Ty_ty SEM_transExp(S_table venv, S_table tenv, A_exp exp)  
{  
    switch (exp->kind) {  
    case (A_opExp):          return SEM_transOpExp(          venv, tenv, exp);  
    case (A_ifExp):          return SEM_transIfExp(          venv, tenv, exp);  
    case (A_forExp):          return SEM_transForExp(          venv, tenv, exp);  
    case (A_letExp):          return SEM_transLetExp(          venv, tenv, exp);  
    case (A_intExp):          return SEM_transIntExp(          venv, tenv, exp);  
    case (A_seqExp):          return SEM_transSeqExp(          venv, tenv, exp);  
    case (A_varExp):          return SEM_transVarExp(          venv, tenv, exp->u.var);  
    case (A_callExp):          return SEM_transCallExp(          venv, tenv, exp);  
    case (A_floatExp):          return SEM_transFloatExp(          venv, tenv, exp);  
    case (A_assignExp):          return SEM_transAssignExp(          venv, tenv, exp);  
    case (A_allocateArrayExp): return SEM_transAllocateArrayExp( venv, tenv, exp);  
    case (A_allocateRecordExp): return SEM_transAllocateRecordExp(venv, tenv, exp);  
    }  
}
```

Semantic Analysis – scanning the AST recursively for errors – SEQ example

```
Ty_ty SEM_transSeqExp(S_table venv, S_table tenv, A_exp exp)
{
    Ty_ty type;
    A_expList expList;

    /***/
    /* [0] Trans exp list */
    /***/
    for (expList=exp->u.seq; expList; expList = expList->tail)
    {
        /***/
        /* [1] Trans the head */
        /***/
        type=SEM_transExp(venv, tenv, expList->head);
    }

    /***/
    /* [2] the returned type of a seq expression is the last expression of the sequence */
    /***/
    return type;
}
```

Semantic Analysis – scanning the AST recursively for errors – IF example

```
Ty_ty SEM_transIfExp(S_table venv, S_table tenv, A_exp exp)
{
    Ty_ty type;

    /* [0] Make sure the expression is indeed an "if" expression */
    assert(exp->kind == A_ifExp);

    /* [1] Check the test expression to make sure it has an integer type */
    type = SEM_transExp(venv, tenv, exp->u.iff.test);
    if (type != Ty_Int())
    {
        EM_error(exp->pos, "condition in IF expression is not an integer");
        return NULL;
    }

    /* [2] trans the result expression */
    type = SEM_transExp(venv, tenv, exp->u.iff.then);

    return type;
}
```


Semantic Analysis – scanning the AST recursively for errors – ASSIGN example

```
Ty_ty SEM_transAssignExp(S_table venv, S_table tenv, A_exp exp)
{
    Ty_ty type1;
    Ty_ty type2;

    /* [0] Make sure the expression is indeed an "assignment" expression */
    assert(exp->kind == A_assignExp);

    /* [1] trans initialized expression */
    type1 = SEM_transExp(venv, tenv, exp->u.assign.exp);

    /* [2] trans variable */
    type2 = SEM_transVarExp(venv, tenv, exp->u.var);

    /* [3] type check */
    if (type1 != type2)
    {
        EM_error(exp->pos, "Illegal assignment\n");
    }

    return type1;
}
```

Semantic Analysis – Symbol Table & Scopes

- מה יודפס בקטע הקוד הבא?

```
int moish=300;

void main(void)
{
    char moish=200;

    for (int i=0;i<1;i++)
    {
        float moish=100;

        printf("%d\n",(int) moish);
    }
}
```

- צדקתם!

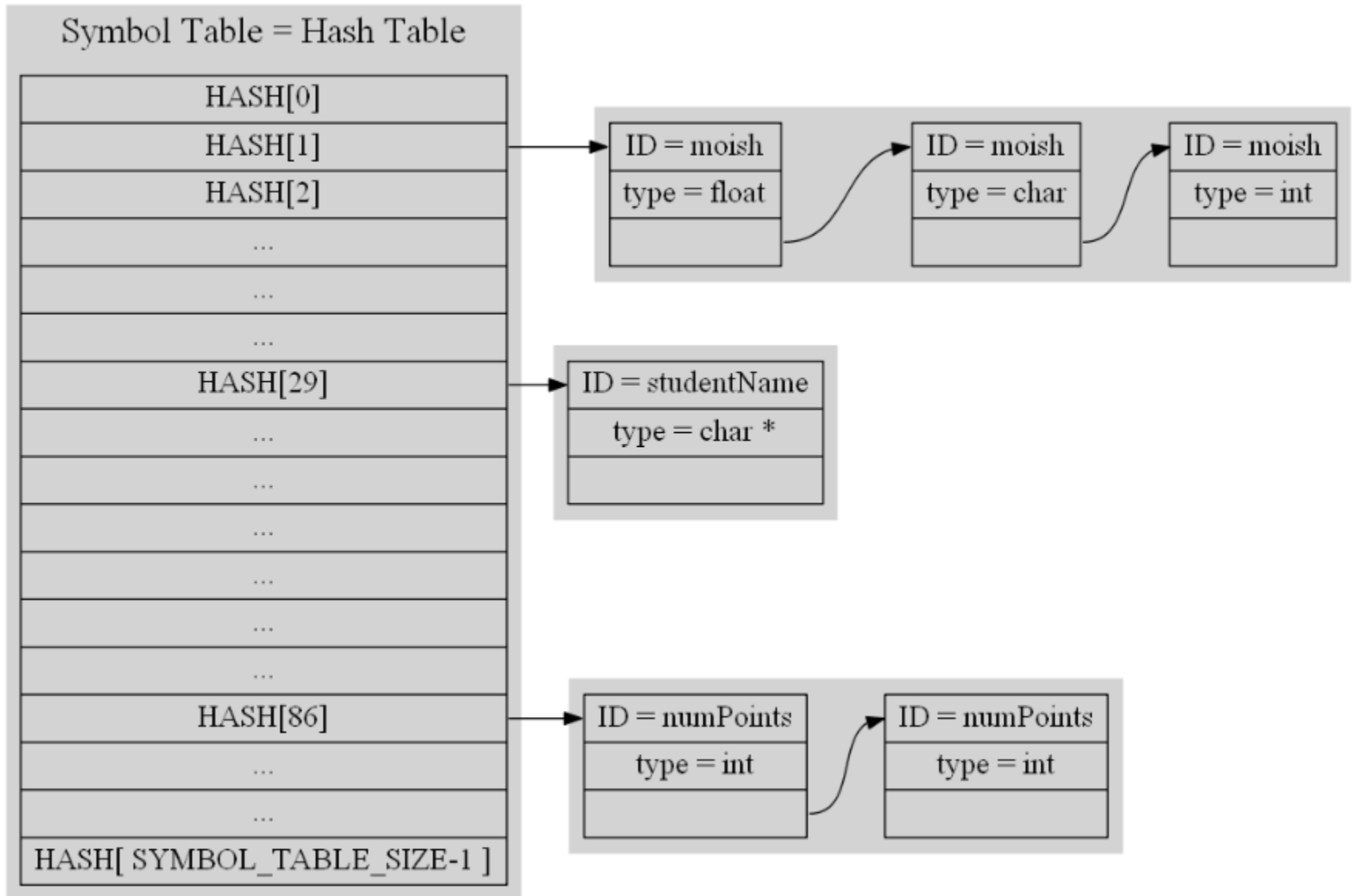


A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Windows\system32\cmd.exe'. The window contains the output of the C program: the number '100' followed by the text 'Press any key to continue . . .'. The window has standard Windows controls (minimize, maximize, close) in the top right corner and a scrollbar on the right side.

Semantic Analysis – Symbol Table & Scopes

- אנחנו רוצים לממש מבנה נתונים שיכיל צמדים (bindings) של משתנה יחד עם הטיפוס שלו.
- מבנה הנתונים צריך לתמוך בפעולות הבאות:
 - "התחל scope חדש"
 - "הכנס binding חדש" (למשל: $\text{moish} \mapsto \text{int}$)
 - "צא מה scope הנוכחי" (שמשמעותו להוציא את כל bindings שהוכנסו ב scope)
 - "מהו ה binding של המשתנה"

Semantic Analysis – Symbol Table & Scopes



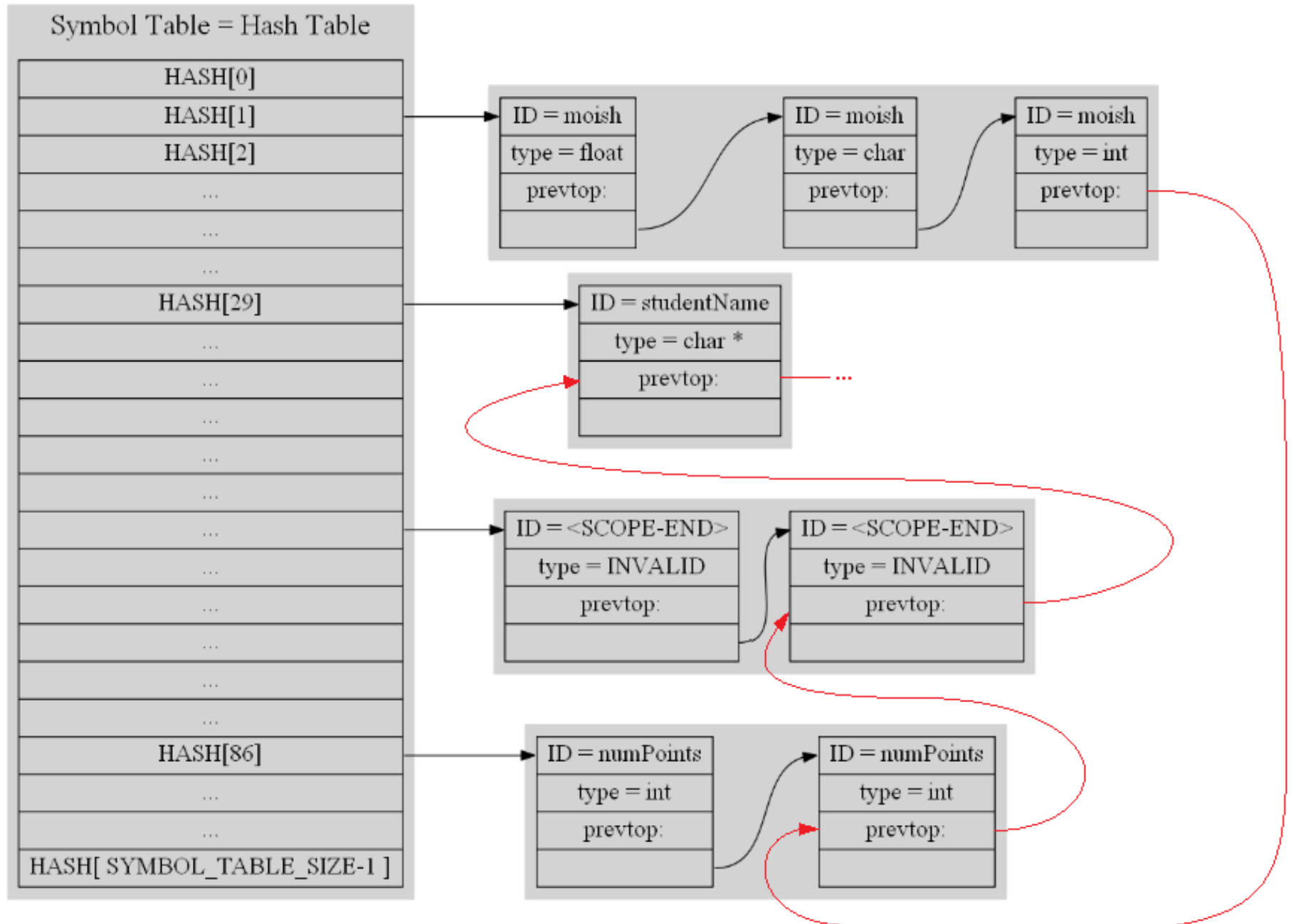
Semantic Analysis – Implementation of Symbol Tables & Scopes

- אבל איך בעצם נממש את היכולת להיכנס ל scope חדש? ולצאת ממנו?
- בתוך טבלת ה hash נממש מחסנית, שתשמור את סדר הכנסת האיברים.
- כשניכנס ל scope חדש, נדחוף למחסנית איבר מיוחד שיסמן את ההתחלה, ונמשיך להכניס לטבלת ה hash כרגיל.
- כשנצא, נתחיל לעשות pop-ים מטבלת ה hash עד שנגיע לאיבר המיוחד שיסמן את זה שהוצאנו את כל משתני ה scope !

Semantic Analysis – Implementation of Symbol Tables & Scopes

- אם נשתמש באותה דוגמא שמסקף 38, ונניח שב scope האחרון נמצאים המשתנים moish, ו numPoints אז בשקף הבא אפשר לראות את האיבר שמסמן התחלת scope ואת ההצבעות ממנו ואליו.

Semantic Analysis – Symbol Table Revised



Semantic Analysis – Symbol Table Code

- הקוד שתומך במחסנית ה scope-ים הוא פשוט למדי:

```
static struct S_symbol_ marksym = {"<mark>",0};

void S_beginScope(S_table t)
{
    S_enter(t,&marksym,NULL);
}

void S_endScope(S_table t)
{
    S_symbol s;
    do s = (S_symbol) TAB_pop(t);
    while (s != &marksym);
}
```


Semantic Analysis – Symbol Table - Symbols

- אפשר לראות ששמות של משתנים ופונקציות מיוצגים ע"י symbol ולא סתם ע"י המחרוזת שלהן.
- כל המופעים של אותו שם יהיו אותו symbol
- זה קורה במהלך בניית ה AST

LetExp:	LET declarations IN exp END ;	{\$.gval.exp = A_LetExp(EM_tokPos,\$2.gval.decList,\$4.gval.exp);
lvalue:	ID lvalue ARROW ID lvalue LBRACK exp RBRACK	{\$.gval.var = A_SimpleVar(EM_tokPos,S_Symbol(\$1.gval.sval));} {\$.gval.var = A_FieldVar(EM_tokPos,\$1.gval.var,S_Symbol(\$3.gval.sval));} {\$.gval.var = A_SubscriptVar(EM_tokPos,\$1.gval.var,\$3.gval.exp);}
AssignExp:	lvalue ASSIGN exp lvalue ASSIGN AllocateArrayExp lvalue ASSIGN AllocateRecordExp	{\$.gval.exp = A_AssignExp(EM_tokPos,\$1.gval.var,\$3.gval.exp);} {\$.gval.exp = A_AssignExp(EM_tokPos,\$1.gval.var,\$3.gval.exp);} {\$.gval.exp = A_AssignExp(EM_tokPos,\$1.gval.var,\$3.gval.exp);}
CallExp:	ID LPAREN RPAREN ID LPAREN ListExpComma RPAREN	{\$.gval.exp = A_CallExp(EM_tokPos,S_Symbol(\$1.gval.sval),NULL, {\$.gval.exp = A_CallExp(EM_tokPos,S_Symbol(\$1.gval.sval),\$3.gval.exp);}

Semantic Analysis – Implementation

```
struct E_entry_
{
    envEntryType kind;

    union
    {
        struct {F_access access; Ty_ty type;} var;

        struct
        {
            F_frame frame;
            Ty_tyList formals;
            Ty_ty result;
        }
        fun;
    }
    u;
};
```

- ה struct משמאל הוא כרגיל union של שתי אפשרויות: משתנה או פונקציה. שימו לב שמחזקים באדום שדות שישמשו אותנו בהמשך הקומפילציה אך אינם רלבנטיים כרגע

Semantic Analysis – Implementation

```
Ty_ty SEM_transLetExp(S_table venv,S_table tenv,A_exp exp)
```

```
{  
    Ty_ty type;  
  
    /* [0] Make sure the expression is indeed a "let" expression */  
    assert(exp->kind == A_letExp);  
  
    /* [1] open a new variable and type environments */  
    S_beginScope(venv);  
    S_beginScope(tenv);  
  
    /* [2] Scan the declarations part */  
    SEM_transDecs(venv,tenv,exp->u.let.decs);  
  
    /* [3] Scan the expressions part */  
    type=SEM_transExp(venv,tenv,exp->u.let.body);  
  
    /* [4] close the new variable and type environments */  
    S_endScope(venv);  
    S_endScope(tenv);  
  
    return type;  
}
```

- בביטוי let נפתחים שני scope-ים: אחד עבור משתנים ופונקציות (venv) והשני עבור טיפוסים (tenv)
- בשלב הבא ה declarations מנותחים, ואז הגוף של ה let המשתנים, הפונקציות והטיפוסים תקפים לגוף ה let בלבד ולא אחריו, ולכן בסיום ניתוח הגוף הסביבות הנ"ל נסגרות

Semantic Analysis – Implementation

- הפונקציה שמטפלת ב declarations היא פשוט switch-case על משתנים, פונקציות וטיפוסים:

```
void SEM_transDecs(S_table venv, S_table tenv, A_decList decList)
{
    /******
    /* [1] trans dec the head */
    /******
    switch (decList->head->kind) {
    case (A_varDec):      SEM_transVarDec( venv,tenv,decList->head); break;
    case (A_typeDec):     SEM_transTypeDec(venv,tenv,decList->head); break;
    case (A_functionDec): SEM_transFuncDec(venv,tenv,decList->head); break;
    }

    /******
    /* [2] trans dec the tail */
    /******
    if (decList->tail != NULL) SEM_transDecs(venv,tenv,decList->tail);
}
```

Semantic Analysis – function declaration

```

/*****
/* [0] check type of the returned value in case it is defined */
*****/
if (f->result != NULL)
{
    resultType = (Ty_ty) S_look(tenv,f->result);
    if (resultType == NULL)
    {
        EM_error(f->pos,"Unknown returned type in function %s\n",f->name);
    }
}

/*****
/* [1] check types of function parameters */
*****/
for (functionParameters = f->params;functionParameters;functionParameters=functionParameters->tail)
{
    type = (Ty_ty) S_look(tenv,functionParameters->head->typ);
    if (type == NULL)
    {
        EM_error(f->pos,"unknown parameter type in function declaration\n",f->name);
    }
}

```

```

Ty_ty SEM_transVarExp(S_table venv,S_table tenv,A_var var)
{
    Ty_ty type;
    Ty_fieldList fieldList;

    switch (var->kind) {
    case (A_simpleVar):

        /******
        /* [0] check if type exists */
        /******
        if (S_look(venv,var->u.simple) == NULL)
        {
            /******
            /* ERROR: type doesn't exist */
            /******
            EM_error(var->pos,"undefined variable %s\n",S_name(var->u.simple));
            return NULL;
        }

        return (Ty_ty) S_look(venv,var->u.simple);

```

let

```
type IntArray = array of int

type address = {ZIP:int, Apartment:int}

type citizen = {BirthYear:int, lastYearSalaries:IntArray, MyAddress:address, ID:int}

var ZIP:citizen := citizen{1976,nil,nil,33546}
```

in

```
(
  ZIP->lastYearSalaries := int[12] of 7800;

  ZIP->lastYearSalaries[7] := 3;

  PrintInt(ZIP->lastYearSalaries[7] + ZIP->lastYearSalaries[6]);

  ZIP->MyAddress := address{69207,19};

  PrintInt(ZIP->MyAddress->ZIP + ZIP->MyAddress->Apartment);

  PrintInt(ZIP->ID)
)
```

end

קריאה רקורסיבית על המשתנה עצמו

Name	Value	Type
fieldList	0x00638190 {head=0x006380f0, A_fieldLi	
head	0x006380f0 {PrintMyNoc A_field_	
PrintMyNodeSerialN	3	int
PrintTheKindOfTree	0x006380f4 "field" Q	char[100
field_name	0x006380b8 {serial_numl S_symbc	
serial_number	3	int
name	0x00637e78 "ZIP" Q	char *
next	0x00000000 <NULL>	S_symbc
field_type_name	0x00637cb0 {serial_numl S_symbc	
pos	77	int
tail	0x00638018 {head=0x006380f0, A_fieldLi	
PrintMyNodeSerialNurr	4	int
PrintTheKindOfTreeIA	0x0063819c "Type\l Q	char[100

Compiler (Global Scope)

```
A_dec A_RecordTypeDec(A_pos pos, S_symbol declared_record_type_name)
{
    A_dec p = (A_dec) checked_malloc(sizeof(*p));
}
```

109 %

Input.txt

```
let

type IntArray = array of int

type address = {ZIP:int, Apartment:int}

type citizen = {BirthYear:int, lastYearSalaries:IntArray, MyAddress:address}

var ZIP:citizen := citizen{1976,nil,nil,33546}

in

(
    ZIP->lastYearSalaries := int[12] of 7800;
```


Name	Value	Type
var_name	0x006380b8 {serial_number	S_symbol_*
serial_number	3	int
name	0x00637e78 "ZIP"	char *
next	0x00000000 <NULL>	S_symbol_*

Compiler (Global Scope)

```

A_dec A_VarDec(A_pos pos, S_symbol var_name, S_symbol type_name, A_
{ ≤ 1ms elapsed
  A_dec p = (A_dec) checked_malloc(sizeof(*p));

```

109 %

Input.txt

let

type IntArray = array of int

type address = {ZIP:int, Apartment:int}

type citizen = {BirthYear:int, lastYearSalaries:IntArray, MyAddress:address}

var ZIP:citizen := citizen{1976,nil,nil,33546}

in

(

ZIP->lastYearSalaries := int[12] of 7800;

ZIP->lastYearSalaries[7] := 3;

משתנה מסוג גישת שדה

var	0x003ab9b8 {kind=A_f A_var_*
kind	A_fieldVar (1) VarType
PrintMyNode	30 int
PrintTheKind	0x003ab9c0 "FIELD" char[100]
pos	231 int
u	{simple=0x003ab8d0 { <unnamed-



```
Ty_ty SEM_transVarExp(S_table venv, S_table tenv, A_var var)
{
    Ty_ty type;
    Ty_fieldList fieldList;

    switch (var->kind) {
    case (A_simpleVar):

        /* [0] check if type exists */
        if (S_look(venv, var->u.simple) == NULL)
```

109 %

Input.txt

in

(

ZIP->lastYearSalaries := int[12] of 7800;

ZIP->lastYearSalaries[7] := 3;

קריאה רקורסיבית על המשתנה עצמו

```
Ty_ty SEM_transVarExp(S_table venv, S_table tenv, A_var var)
{
    Ty_ty type;
    Ty_fieldList fieldList;

    switch (var->kind) {
    case (A_simpleVar):

        /******
        /* [0] check if type exists */
        /******
        if (S_look(venv, var->u.simple) == NULL)
        {
            /******
            /* ERROR: type doesn't exist */
            /******
            EM_error(
                var->pos,
                "undefined variable %s\n",
                S_name(var->u.simple));
        }

        return (Ty_ty) S_look(venv, var->u.simple);

    case (A_fieldVar):

        type = SEM_transVarExp(venv, tenv, var->u.field.var); ≤ 2ms elapsed
```

השדה ZIP ביצירת AST Node מסוג

הגדרת רשומה

me	Value	Type
fieldList	0x00338190 {head=0x003380f0	A_fieldList_*
head	0x003380f0 {PrintMyNodeSerial	A_field_*
PrintMyNodeSerial	3	int
PrintTheKindOfTree	0x003380f4 "field"	char[100]
field_name	0x003380b8 {serial_number=3	S_symbol_*
serial_number	3	int
name	0x00337e78 "ZIP"	char*
next	0x00000000 <NULL>	S_symbol_*
field_type_name	0x00337cb0 {serial_number=0	S_symbol_*
pos	77	int
tail	0x00338018 {head=0x00337f78	A_fieldList_*
PrintMyNodeSerialNu	4	int
PrintTheKindOfTreeIA	0x0033819c "Type\nFields	char[100]

```
Compiler (Global Scope)

A_dec A_RecordTypeDec(A_pos pos, S_symbol declared_record_type_
{
  A_dec p = (A_dec) checked_malloc(sizeof(*p));

109 %

Input.txt
let

  type IntArray = array of int

  type address = {ZIP:int, Apartment:int}

  type citizen = {BirthYear:int, lastYearSalaries:IntArray, MyAc

  var ZIP:citizen := citizen{1976,nil,nil,33546}

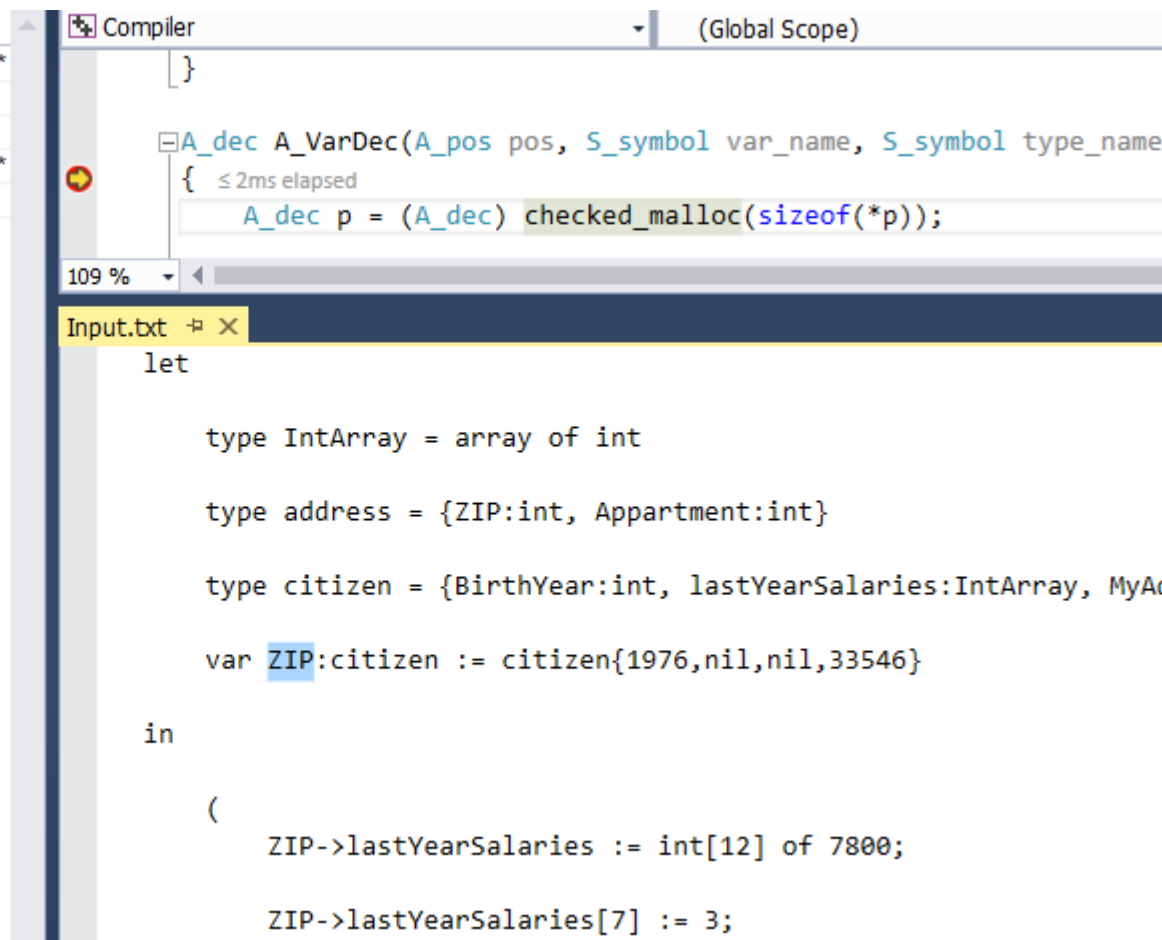
in

(
  ZIP->lastYearSalaries := int[12] of 7800;

  ZIP->lastYearSalaries[7] := 3;
```

המשתנה ZIP ביצירת AST Node מסוג הגדרת משתנה

me	Value	Type
var_name	0x003380b8 {serial_number=3	S_symbol_*
serial_number	3	int
name	0x00337e78 "ZIP"	char *
next	0x00000000 <NULL>	S_symbol_*



The screenshot shows a compiler window with a function call `A_dec A_VarDec(A_pos pos, S_symbol var_name, S_symbol type_name)` and a code editor with the following code:

```
let  
  
    type IntArray = array of int  
  
    type address = {ZIP:int, Apartment:int}  
  
    type citizen = {BirthYear:int, lastYearSalaries:IntArray, MyA:  
  
    var ZIP:citizen := citizen{1976,nil,nil,33546}  
  
in  
  
    (  
        ZIP->lastYearSalaries := int[12] of 7800;  
  
        ZIP->lastYearSalaries[7] := 3;
```

כניסה לבדיקת משתנה

Name	Value	Type
var	0x0033b9b8 {kind=A_fieldV; A_var_*	A_var_*
kind	A_fieldVar (1)	VarType
PrintMyNo	30	int
PrintTheKir	0x0033b9c0 "FIELD\nV" Q	char[100]
pos	231	int
u	{simple=0x0033b8d0 {serial_ <unnamed-tag>	<unnamed-tag>

Compiler (Global Scope)

```
Ty_ty SEM_transVarExp(S_table venv, S_table tenv, A_var var)
{
    ≤ 11ms elapsed
    Ty_ty type;
    Ty_fieldList fieldList;

    switch (var->kind) {
    case (A_simpleVar):
```

109 %

Input.txt

```
in
(
    ZIP->lastYearSalaries := int[12] of 7800;
    ZIP->lastYearSalaries[7] := 3;
```

קריאה רקורסיבית לבדיקת משתנה

```
Ty_ty SEM_transVarExp(S_table venv, S_table tenv, A_var var)
{
    Ty_ty type;
    Ty_fieldList fieldList;

    switch (var->kind) {
    case (A_simpleVar):

        /******
        /* [0] check if type exists */
        /******
        if (S_look(venv, var->u.simple) == NULL)
        {
            /******
            /* ERROR: type doesn't exist */
            /******
            EM_error(
                var->pos,
                "undefined variable %s\n",
                S_name(var->u.simple));
        }

        return (Ty_ty) S_look(venv, var->u.simple);

    case (A_fieldVar):

        type = SEM_transVarExp(venv, tenv, var->u.field.var); ≤ 3ms elapsed
```

האם המשתנה ZIP קיים?

Name	Value	Type
var	0x0033b8d0 {kind=A_sim	A_var_*
kind	A_simpleVar (0)	VarType
PrintMyNodeSer	29	int
PrintTheKindOfT	0x0033b8d8 "SIMPLE Q	char[100]
pos	229	int
u	{simple=0x003380b8 {seri	<unnamed-tag>
simple	0x003380b8 {serial_num	S_symbol_*
serial_num	3	int
name	0x00337e78 "ZIP"	char *
next	0x00000000 <NULL>	S_symbol_*
field	{var=0x003380b8 {kind=A	<unnamed-tag>
subscript	{var=0x003380b8 {kind=A	<unnamed-tag>

Compiler (Global Scope)

```
Ty_ty SEM_transVarExp(S_table venv, S_table tenv, A_var var)
{
    Ty_ty type;
    Ty_fieldList fieldList;

    switch (var->kind) {
    case (A_simpleVar):

        /******
        /* [0] check if type exists */
        /******

        if (S_look(venv, var->u.simple) == NULL) ≤ 4ms elapsed
        {
            /******
            /* ERROR: type doesn't exist */
            /******
            EM_error(
                var->pos,
                "undefined variable %s\n",
                S_name(var->u.simple));
        }
    }
```


הסימבול (היחיד!) שמכיל את המחרוזת ZIP.

הכתובת של ZIP היא כתובת ההקצאה הראשונה של המשתנה מהלקסר

The screenshot shows a debugger interface with a variable window on the left and a code window on the right.

Variable Window:

var->u.simple	0x003380b8 {serial_number	S_symbol_*
serial_number	3	int
name	0x00337e78 "ZIP"	char*
next	0x00000000 <NULL>	S_symbol_*

Code Window:

```
Ty_ty SEM_transVarExp(S_table venv, S_table tenv, A_var var)
{
    Ty_ty type;
    Ty_fieldList fieldList;

    switch (var->kind) {
    case (A_simpleVar):

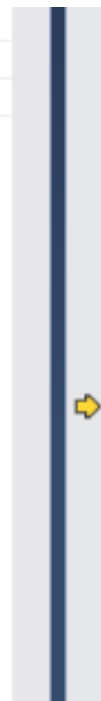
        /******
        /* [0] check if type exists */
        /******
        if (S_look(venv, var->u.simple) == NULL)
        {
            /******
            /* ERROR: type doesn't exist */
            /******
            EM_error(
                var->nos
```

Call Stack:

Name
Compiler.exe!TAB_look(TAB_table_ *t, void *key) Line 115
Compiler.exe!S_look(TAB_table_ *t, S_symbol_ *sym) Line 98
Compiler.exe!SEM_transVarExp(TAB_table_ *venv, TAB_table_ *tenv, A_var_ *var) Line 722

אכן מפתח מוזר ל hash ... מתייחסים לכתובת ההקצאה כאל (unsigned int) ועליו עושים hash

key,x	0x003380b8	void *
index	9	int



```
void *TAB_look(TAB_table t, void *key)
{
    int index;
    binder b;
    assert(t && key);

    index=((unsigned)key) % TABSIZE;

    for (b=t->table[index]; b; b=b->next) ≤ 5ms elap:
    {
        if (b->key==key)
        {
            return b->value;
        }
    }
}
```

האם הטיפוס שחזר הוא מסוג רשומה?

Name	Value	Type
type	0x00341958 {kind	Ty_ty_*
kind	Ty_record (7)	TypeKind
u	{name=0x003419	<unnamed-ta
name	0x00341920 {kind	Ty_ty_*
array	0x00341920 {kind	Ty_ty_*
record	0x00341920 {head	Ty_fieldList_*

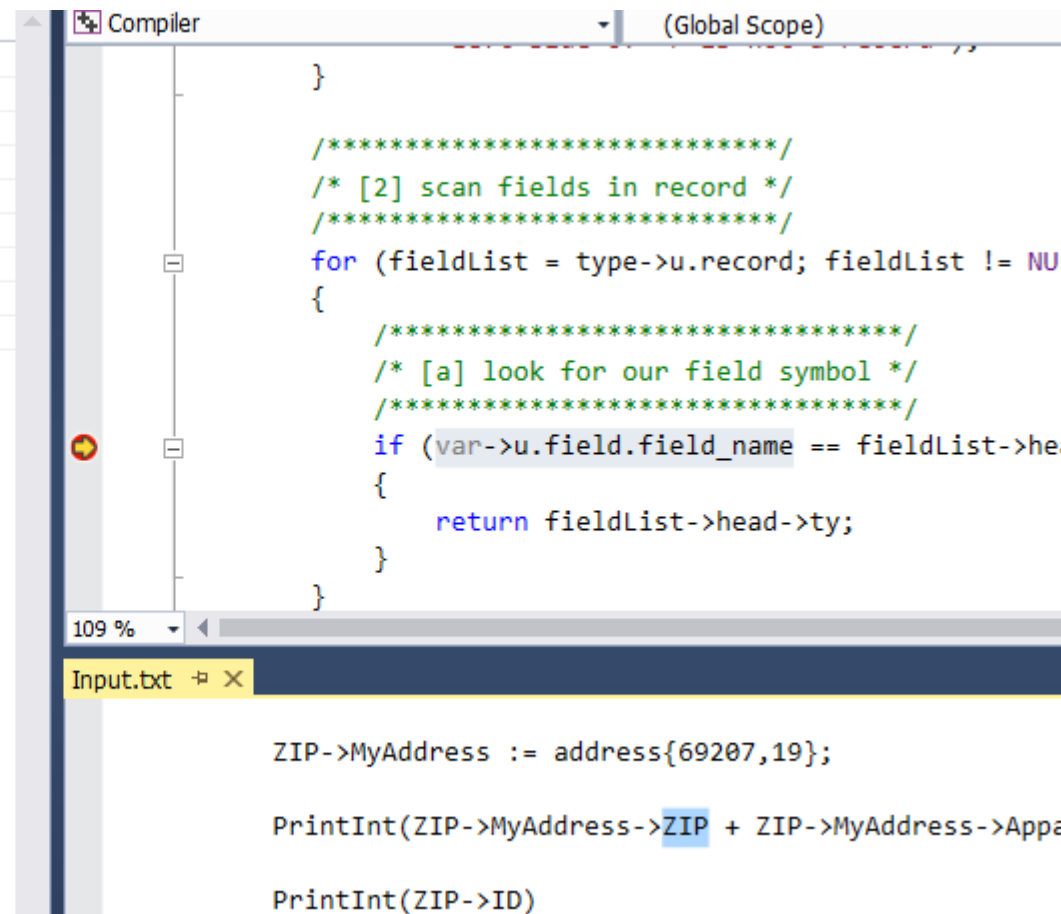
Compiler

(Global Scope)

```
case (A_fieldVar):  
  
    type = SEM_transVarExp(venv,tenv,var->u.field.var);  
  
    /* [1] check if type is a record */  
    if (type->kind != Ty_record) ≤ 2ms elapsed  
    {  
        EM_error(  
            var->pos,  
            "Left side of -> is not a reocrd");  
    }  
}
```

השוואה בין מחרוזות היא בעצם השוואה בין
הסימבולים היחידים שמכילים את המחרוזות.
כלומר, השוואה בין כתובות (32 ביט)

name	Value	Type
fieldList->head->name	0x003380b8 {serial_number=3 name="ZIP" next=0x00000000 <NULL>}	S_symbol_*
serial_number	3	int
name	0x00337e78 "ZIP"	char*
next	0x00000000 <NULL>	S_symbol_*
var->u.field.field_name	0x003380b8 {serial_number=3 name="ZIP" next=0x00000000 <NULL>}	S_symbol_*
serial_number	3	int
name	0x00337e78 "ZIP"	char*
next	0x00000000 <NULL>	S_symbol_*



```
Compiler (Global Scope)
}
/* [2] scan fields in record */
for (fieldList = type->u.record; fieldList != NULL; fieldList = fieldList->next)
{
    /* [a] look for our field symbol */
    if (var->u.field.field_name == fieldList->name)
    {
        return fieldList->head->ty;
    }
}

109 %
Input.txt
ZIP->MyAddress := address{69207,19};
PrintInt(ZIP->MyAddress->ZIP + ZIP->MyAddress->Appa
PrintInt(ZIP->ID)
```

Templates

```
template <class T, class S>
void TRUMP(T i, S j)
{
    printf("%d\n", (int) i+j);
}

int main(void)
{
    TRUMP(5.0, 4.0);
    TRUMP(5, 4.0);
    TRUMP(5.0, 4);
}
```