

# מבוא לקומפילציה

## חלק 2

### מגישים:

מתן דוידאן 205509219

תום דמרי 205770068

אמיר דאוד 204492292

#### LEX AND YACC VERSIONS:

```
tom@tom:~$ yacc -V
yacc - 1.9 20140715

tom@tom:~$ yacc -V
yacc - 1.9 20140715

tom@tom:~$ lex -V
flex 2.6.4

tom@tom:~$
```

בחוברת זו נציג, את הדוגמאות לחלק 2 .

לכל סעיף עשינו **לפחות** שני חלקים, חלק עובד וחלק לא עובד – קודם נציג את כל הבדיקות ואח"כ את התוצאות.

עשינו סקריפט שרץ על כל קבצי הבדיקות.

קובץ ששמו הוא: XZnwY.t הוא קובץ בדיקה לסעיף מס' X שהתוצאה שלו היא שגיאה.

קובץ ששמו הוא: XZwY.t הוא קובץ בדיקה לסעיף מס' X שהתוצאה שלו היא יצירת עץ (נציג את העצים).

Z,Y מייצגים תתי סעיפים.

במקום Z וY יכול להופיע כיתוב נוסף, משום שלא החלטנו על איך לקרוא לסעיפים עם תתי סעיפים.

1a-w.

```
function
int
foo()
{
    var int x;
    {
        var int y;
        x = 1;
        y = 2;
        {
            x = 2;
        }
        y = 3;
    }
    return 0;
}
function void main()
{
    {
        {} /* empty code blocks are okay, although not very useful */
    }
}
```

1a-nw.

```
function int foo()
{
    var int x;
    {
        var int y;
        x = 1;
        y = 2;
        {
            x = 2;
        }
        y = 3;
    }
    return 0;
}
function int foo2()
{
    {
        {} /* empty code blocks are okay, although not very useful */
    }
    return 0;
}
```

1b-nw.

```
function
int
foo(int
i, j,
k){

    function int foo(int i, j, k)
    {
        function void main(){
        }

    }
    string a[30], b[100] = "moshe";
    var char c;
    c = 'e'; /* everything up to this is OK */
    c = a; /* type mismatch, can't assign string type to character type
    */
    a[0] = 'e';
    /* cannot add anything to array elements - they are not pointers */

}
```

1b-w.

```
function
void
main()
{
    {
        {}
    }
}
function int foo()
{
    var int x;
    {
        var int y;
        x = 1;
        y = 2;
        {
            x = 2;
        }
        y = 3;
    }
    return 0;
}
```

1c-nw.

```
function int foo() { return 0; }  
function int foo_2() { var int a; a = 2; return a; }  
function int foo_3() { if (true) { return foo(); } return 0; }  
function void foo_4() { var int a; a = 2; }
```

1d-nw.

```
function  
void  
foo()  
  
    {  
        var int a = 5;  
    }
```

2a-nw.

```
function
int
foo()
{
    var int x;
    {
        var int y;
        x = 1;
        y = 2;
        {
            x = 2;
        }
        y = 3;
    }
    return 0;
}
function int main()
{
    {
        {} /* empty code blocks are okay, although not very useful */
    }
    return 0;
}
function int foo2()
{
    var int x;
    {
        var int y;
        x = 1;
        y = 2;
        {
            x = 2;
        }
        y = 3;
    }
    return 0;
}
```

2a-w.

```
function
int
foo() {
return
0; }

function int foo_2() { var int a; a = 2; return a; }
function int foo_3() { if (true) { return foo(); } return 0; }
function void main() { var int a; a = 2; }
```

2b-nw.

```
function
int
foo()
{
    var int x;
    {
        var int y;
        x = 1;
        y = 2;
        {
            x = 2;
        }
        y = 3;
    }
    return 0;
}

function void main(int a, b, y; char c)
{
    {
        {} /* empty code blocks are okay, although not very useful */
    }
}
```

3-nw.

```
function
int
foo() {
return
0; }

function int foo_2() { var int a; a = 2; return a; }
function int foo_3() { if (true) { return foo(); } return 0; }
function char foo_2() { var char a; a = 'g'; return a; }
function void main() { var int a; a = 2; }
```

3-w.

```
function
int
foo() {
return
0; }

function int foo_2() { var int a; a = 2; return a; }
function int foo_3() {
    function int foo_2() { var int a; a = 2; return a; }/*function with
same name in other scope*/
    if (true) { return foo(); }
    return foo_2();
}
function void main() { var int a; a = 2; }
```



4a-nw.

```
function
int
foo(){

    var int x;
    var int* y;
    var char* x;
    string y[10];
    var char x;      /* x redaclared */
    x = 5;
    y = &x;
    x = 6;

    y = "foobar";
    x = *(x - 5);
    y = "barfoo";
}
```

4a-w.

```
function
int
foo(){

    var int x;
    var int* l;
    var char* k;
    string y[10];
    var char z;

    x = 5;

    y = "foobar";
    k = &y[5]; /* k points to 'r' */
    z = *(k - 5); /* z is 'f' */
    y = "barfoo"; /* z is still 'f', but x now points to 'o' */
    return *l;
}
function void main(){}

```

4b-nw.

```
function int foo(){
    var int x;
    var int* y;
    var char* x;
    string y[10];
    var char z;
    x = 5;
    y = &x;
    x = 6;
    {
        var int x; /*redaclaration in other scope. */
        var int* y;
        var char* x;
        string y[10];
        var char z;
        x = 5;
        y = &x;
        x = 6;
    }

    y = "foobar";
    x = &y[5]; /* x points to 'r' */
    z = *(x - 5); /* z is 'f' */
    y = "barfoo"; /* z is still 'f', but x now points to 'o' */
}
```

5-w.

```
/* long
comment long
comment */

function bool foo1(int a, b, y; char c)
{
    var real s;
    var int* k=null,m;
    var int g,l;
    var string p;
    var bool res = true;
    var char* f,i;
    s = 4.5;
    c=*f;
    i = f-7;
    c = c;

    {
        var char x = 'd', b;
        var int y;
        b = '&';
        a = (y * 7)/a-y;
        res = (b == c) && (y > a);
    }
    return res ;
}

function void fee1(int i, j, k, x)
{
    function bool fee2(int l, m, n)
    {
        var bool x, j; /* declarations */
        var char* k;
        *k = '@'; /* statements */
        i = l + l;
        if ((*k == '*') || (false != false) && (l + m < i))
        {
            x = l < m;
        }
        return x;
    }
}
```

```

    var bool ghgh; /* declarations */
    {
        var char x;
        var bool k = true;
        k = foo1(5,i, 34,x);
    }
    x = k;
}

```

```

function int foo2()
{
    string s1[100]="aba", s2[3], s3[20]="4";
    var int i, j=0, cnt;
    string a[30], b[100] = s3;
    var char c;
    var int ds =i;
    c = 'e';
    a[19] = 'f';
    a[4+2] = 'g';
    b = a;
    b[3] = c;
    a = "test"; /* basically equivalent to a[0] = 't'; a[1] = 'e'; a[2] =
    's'; a[3] = 't'; a[4] = '\0'; */
    i = |b|; /* this assigns 100 to variable i, since the length
    operator returns the size of the character array */
    cnt = 1;
    for(i=0; i<|s1|; i=i+1){
        do{
            if (s1[i] == s2[j])
                cnt = cnt*2;
            j=j+1;
        }while(false && true);
        j=i*2;
    }
    return cnt;
}

function void main()
{
    var int a;
    a = foo2();
}

```

```

function int foo4()

```

```

{
    do{
        var int j;
        j=j+1;
    }while(true);
{
    var int x,k;
    var int* y;
    x = 5;
    y = &x;
    x = 6;
    if (&x == y && *y == x)
        *y = 9;
    {
        var char* x;
        string y[10];
        var char z;
        y = "foobar";
        x = &y[5*3+5*k];
        z = *(x + 5);
        y = "barfoo";
    }
}
return 0;
}

function int foo10()
{
    function int foo20()
    {

    }
    function int foo30()
    {

    }
    function int foo40()
    {
        function int foo41()
        {
            var char i;
        }
        function bool foo42()
        {
            var int i, j, k,x;
        }
    }
}

```

```
function void foo43()
{
    var int i, j, k,x,l,m;
    var bool df;
    if (fee1(i, j, k, x) == foo43())
    {
        df = 1 < m;
    }
}

}
```

5a-nw.

```
/*
long
comment long
comment */

function bool foo1(int a, b, y; char c)
{
    var real s;
    var int* k=null,m;
    var int g,l;
    var string p;
    var bool res = true;
    var char* f,i;
    s = 4.5;
    c=*f;
    i = f-7;
    c = c;

    {
        var char x = 'd', b;
        var int y;
        b = '&';
        a = (y * 7)/a-y;
        res = (b == c) && (y > a);
    }
    return res ;
}

function void fee1(int i, j, k, x)
{
    function bool fee2(int l, m, n)
    {
        var bool x, j; /* declarations */
        var char* k;
        *k = '@'; /* statements */
        i = l + l;
        if ((*k == '*') || (false != false) && (l + m < i))
        {
            x = l < m;
        }
        return x;
    }
    var bool ghgh; /* declarations */
    {
```

```

        var char x;
        var bool k = true;
        k = foo1(5,i, 34,x);
    }
    x = k;
}

```

```

function int foo2()
{
    string s1[100]="aba", s2[3], s3[20]="4";
    var int i, j=0, cnt;
    string a[30], b[100] = s3;
    var char c;
    var int ds =i;
    c = 'e';
    a[19] = 'f';
    a[4+2] = 'g';
    b = a;
    b[3] = c;
    a = "test"; /* basically equivalent to a[0] = 't'; a[1] = 'e'; a[2] =
    's'; a[3] = 't'; a[4] = '\0'; */
    i = |b|; /* this assigns 100 to variable i, since the length
    operator returns the size of the character array */
    cnt = 1;
    for(i=0; i<|s1|; i=i+1){
        do{
            if (s1[i] == s2[j])
                cnt = cnt*2;
            j=j+1;
        }while(false && true);
        j=i*2;
    }
    return cnt;
}

```

```

function int foo4()
{
    do{
        var int j;
        j=j+1;
    }while(true);
    {
        var int x,k;
    }
}

```



```

    var int* y;
    x = 5;
    y = &x;
    x = 6;
    if (&x == y && *y == x)
        *y = 9;
    {
        var char* x;
        string y[10];
        var char z;
        y = "foobar";
        x = &y[5*3+5*k];
        z = *(x + 5);
        y = "barfoo";
    }
}
return 0;
}

```

```

function void main()
{
    var int a;
    a = foo2();
}

```

```

function int foo10()
{
    function int foo20()
    {
        foo43(); /* call before declaration */
    }
    function int foo30()
    {

    }
    function int foo40()
    {
        function int foo41()
        {
            var char i;
        }
        function bool foo42()
        {
            var int i, j, k,x;

```

```
    }  
    function void foo43()  
    {  
        var int i, j, k,x,l,m;  
        var bool df;  
        if (feel(i, j, k, x) == foo43())  
        {  
            df = 1 < m;  
        }  
    }  
}  
  
}
```

5b-nw.

```
/* long
comment long
comment */

function bool foo1(int a, b, y; char c)
{
    var real s;
    var int* k=null,m;
    var int g,l;
    var string p;
    var bool res = true;
    var char* f,i;
    g = foo2(); /* call before declaration */
    s = 4.5;
    c=*f;
    i = f-7;
    c = c;

    {
        var char x = 'd', b;
        var int y;
        b = '&';
        a = (y * 7)/a-y;
        res = (b == c) && (y > a);
    }
    return res ;
}

function void fee1(int i, j, k, x)
{
    function bool fee2(int l, m, n)
    {
        var bool x, j; /* declarations */
        var char* k;
        *k = '@'; /* statements */
        i = l + l;
        if ((*k == '*') || (false != false) && (l + m < i))
        {
            x = l < m;
        }
        return x;
    }
}
```

```

    var bool ghgh; /* declarations */
    {
        var char x;
        var bool k = true;
        k = foo1(5,i, 34,x);
    }
    x = k;
}

```

```

function int foo2()
{
    string s1[100]="aba", s2[3], s3[20]="4";
    var int i, j=0, cnt;
    string a[30], b[100] = s3;
    var char c;
    var int ds =i;
    c = 'e';
    a[19] = 'f';
    a[4+2] = 'g';
    b = a;
    b[3] = c;
    a = "test"; /* basically equivalent to a[0] = 't'; a[1] = 'e'; a[2] =
    's'; a[3] = 't'; a[4] = '\0'; */
    i = |b|; /* this assigns 100 to variable i, since the length
    operator returns the size of the character array */
    cnt = 1;
    for(i=0; i<|s1|; i=i+1){
        do{
            if (s1[i] == s2[j])
                cnt = cnt*2;
            j=j+1;
        }while(false && true);
        j=i*2;
    }
    return cnt;
}

```

```

function int foo4()
{
    do{
        var int j;
        j=j+1;
    }while(true);
}

```

```

{
    var int x,k;
    var int* y;
    x = 5;
    y = &x;
    x = 6;
    if (&x == y && *y == x)
        *y = 9;
    {
        var char* x;
        string y[10];
        var char z;
        y = "foobar";
        x = &y[5*3+5*k];
        z = *(x + 5);
        y = "barfoo";
    }
}
return 0;
}

```

```

function void main()
{
    var int a;
    a = foo2();
}

function int foo10()
{
    function int foo20()
    {

    }

    function int foo30()
    {

    }

    function int foo40()
    {
        function int foo41()
        {
            var char i;
        }
        function bool foo42()

```

```

{
    var int i, j, k,x;
}
function void foo43()
{
    var int i, j, k,x,l,m;
    var bool df;
    if (fee1(i, j, k, x) == foo43())
    {
        df = 1 < m;
    }
}
}
}

```

6-nw.

```
function
void
main(){

    if(3 > 2)
    {
        /*...statements...*/
        i = 5; /* i has not been declared*/
    }

}
```

6a-w.

```
function
void
main(){

    var int i;
    if(3 > 2)
    {
        /*...statements...*/
        i = 5; /* i has not been declared*/
    }

}
```

6b-w.

```
function
void
main(){

    if(3 > 2)
    {
        var int i;
        /*...statements...*/
        i = 5; /* i has not been declared*/
    }

}
```

7-nw.

```
function int foo(int a,b,c)
{
    var int x;
    {
        var int y;
        x = 1;
        y = 2;
    }
    x = 2;
    y = 3;
}
return 0;
}
function int foo2()
{
    var int a,b,c,d,e;
    a= foo(a,b,c,d);
    {
        {} /* empty code blocks are okay, although not very useful */
    }
    return 0;
}
function void main()
{
}
```



7-w.

```
function
int
foo(int
a,b,c)
{
    var int x;
    {
        var int y;
        x = 1;
        y = 2;
    {
        x = 2;
    }
        y = 3;
    }
    return 0;
}
function int foo2()
{
    var int a,b,c,d,e;
    a= foo(a,b,c);
    {
        {} /* empty code blocks are okay, although not very useful */
    }
    return 0;
}
function void main()
{
}
```

8-a-nw.

```
function
int
foo(int
a,b,c)
{
    var int x;
    {
        var int y;
        x = 1;
        y = 2;
        {
            x = 2;
        }
        y = 3;
    }
    return 0;
}
function int foo2()
{
    var int a,b,c;
    var char d,e;
    a= foo(a,b,e);
    {
        {} /* empty code blocks are okay, although not very useful */
    }
    return 0;
}
function void main()
{
}
```

8-w.

```
function
int
foo(int
a,b;
char j)
{
    var int x;
    {
        var int y;
        x = 1;
        y = 2;
        {
            x = 2;
        }
        y = 3;
    }
    return 0;
}
function int foo2()
{
    var int a,b,c;
    var char d,e;
    a= foo(a,b,e);
    {
        {} /* empty code blocks are okay, although not very useful */
    }
    return 0;
}
function void main()
{
}
```

9nw.

```
/*Test if  
the return  
value type  
is the same  
as declared  
in function  
return type  
- should be  
incorrect*/
```

```
function int foo()  
{  
    var char x = 'a';  
    return x;  
}
```

9nw2.

```
/*Function  
return  
value type  
should not  
be a  
string*/
```

```
function string smt()  
{  
    string str[100] = "Test";  
    return str;  
}
```

9w.

```
/*Test if the function return value type is of the same type as the function  
signature type - should be correct*/
```

```
function int foo()  
{  
    var int a = 10;  
    if(a == 10)  
    {  
        a = a + 1;  
    }  
    return a;  
}
```

```
function void main()  
{  
    foo();  
}
```

10nw.

```
/*This test should be incorrect, the function return type value is not the same  
type of the variable that the function value assigned to*/
```

```
function char f2()  
{  
    var char x = 'b';  
    var bool flag = true;  
  
    if (flag != false)  
    {  
        return x;  
    }  
  
    else {  
        return 'd';  
    }  
  
    return 'e';  
}  
  
function void main()  
{  
    var int a = 6;  
    a = f2();  
}
```

10w.

```
/*This test should be correct, the function return type is the same type as the
variable type*/
```

```
function int f1()
{
    var int a = 9;
    var int b = 7;
    if(a>b)
    {
        return b+5;
    }
    else {
        return a*6;
    }

    return 0;
}

function void main()
{
    var int c = 0;
    c = f1();
}
```

11nw.

```
/*This test should be incorrect, the checked type inside the if statement is not of Boolean*/
```

```
function void main()
{
    var int a = 17;
    var int b = 13;
    var int c = a*b/a+b;
    if(c)
    {
        a = b/c*c*a;
    }
    else {
        b = a*2/b*c;
    }
}
```

11w.

```
/*This test should be correct, the checked type inside the if statement is of Boolean*/
```

```
function void main()
{
    var int a = 10;
    var bool check = true;
    if(a == 10)
    {
        if(check)
        {
            a = 2;
        }
    }
}
```



12wfor.

```
/*This test should be correct, the variable type checked in the loop condition  
is of type Boolean*/
```

```
function void t()  
{  
    var bool p = true;  
    var int a = 0;  
    for(a = 10; p; a = a + 1)  
    {  
        if(a>10*3)  
        {  
            p = false;  
        }  
    }  
}  
  
function void main()  
{  
    t();  
}
```

12nwdowhile.

```
/*This test should be incorrect, the variable type checked in the loop condition  
is not of type boolean*/
```

```
function void t()  
{  
    var int p2 = 34;  
    do  
    {  
        p2 = p2 - 1;  
        if(p2<10)  
            p2 = p2 - 3;  
    } while(p2);  
}  
  
function void main()  
{
```

```

        t();
    }

```

12nwfor.

```

/*This test should be incorrect, the variable type checked in the loop condition
is not of type boolean*/

```

```

function void t()
{
    var int a = 0;
    for(a = 12; a; a = a + 1)
    {
        if(a>10*3)
        {
            a = a + 1;
        }
    }
}

function void main()
{
    t();
}

```

12nwwhile.

```

/*This test should be incorrect, the variable type checked in the loop condition
is not of type boolean*/

```

```

function void t()
{
    var real rn = 16.3;
    var int p = 8;
    while(rn)
    {
        if(rn > p * 3)
        {

```

```

        p = p / 2;
    }
}

function void main()
{
    t();
}

```

12wdowhile.

```

/*This test should be correct, the variable type checked in the loop condition
is of type boolean*/

```

```

function void t()
{
    var bool t = true;
    var int c = 10;
    var int d = 100;
    do
    {
        d = d - 10;
        if(c > d)
        {
            t = false;
        }
    } while(t);
}

function void main()
{
    t();
}

```

12wwhile.

```
/*This test should be correct, the variable type checked in the loop condition  
is of type boolean*/
```

```
function void t()  
{  
    var bool p = true;  
    var int a = 7;  
    var int b = 8;  
    while(p)  
    {  
        if(a<b)  
        {  
            p = false;  
        }  
    }  
}  
  
function void main()  
{  
    t();  
}
```

13nw.

```
/*This test should be incorrect, the type of the array index should is not of  
type int*/
```

```
function void main()  
{  
    string str2[100] = "TestingIndexTypeV2";  
    var real t1 = 1.4;  
    var bool p = true;  
    while(p)  
    {  
        str2[t1] = '0';  
        t1 = t1+2*t1;  
        if(t1 > 35.5)  
        {  
            p = false;  
        }  
    }  
}
```

13w.

```
/*This test should be correct, the type of the array index should be an int*/
```

```
function void main()  
{  
    string str1[100] = "TestingIndexType";  
    var int k = 1;  
    var bool t = true;  
  
    while(t)  
    {  
        k = k * 2;  
        str1[k] = 'n';  
        if(k<6*2)  
        {  
            t = false;  
        }  
    }  
}
```

14nw.

```
/*This test should be incorrect, the operator [] is used only on string(char  
array) type*/
```

```
function int f1(int b)
{
    var bool p = true;
    p[2]='s';
    return 1;
}

function void main()
{
    var int v1 = 5;
    var int v2 = f1(v1);
}
```

14nw2.

```
/*This test should be incorrect, the operator [] is used only on string(char  
array) type*/
```

```
function int f1(int b)
{
    var int arr1 = 3;
    arr1[b+1] = 't';
    return b-2;
}

function void main()
{
    var int v1 = 1;
    var int v2 = f1(v1);
}
```

14w.

```
/*This test should be correct, the operator [] is used only on string(char  
array) type*/
```

```
function int f1(int b)  
{  
    string str2[50] = "OperatorTest";  
    str2[b/2] = 'N';  
    return 1;  
}
```

```
function void main()  
{  
    var int v1 = 5;  
    var int v2 = f1(v1);  
}
```

15nw.

*/\*This test should be incorrect, the variable `type` that is left to '=' is `not` the same `type` as the variable right to '='\*/*

```
function void main()
{
    var int a = 4;
    var real b = 7.6;
    a = b*2;
}
```

15w1.

```
function void main()
{
    var int a = 4;
    var int b = 7;
    var int c = a + b;
    var real d = 2.2;
    var real e = 3.1;
    var real f = e;
    var bool c1 = true;
    var bool c2 = c1;
    string str1[10] = "Testing..";
    str1[2] = 'T';
}
```

15w2.

*/\*This test should be correct, you can assign only null to int, real `or` char pointers\*/*

```
function void main()
{
    var char* p = null;
    var int* c = null;
    var real* d = null;
}
```



16a-nw.

```
function void main(){  
  
    var bool e= true;  
    var int a=4;  
    e = !a;  
  
}
```

16a-w.

```
function void main(){  
    var real c = 94.95;  
    var int d = 77,e;  
    e= d+e;  
    c=c+d;  
    c=c+c;  
  
}
```

16b-nw.

```
function void main(){  
    var int a=4,b;  
    b=|a|;  
  
}
```

16b-w.

```
function void main(){  
    var bool a=true,b=false;  
    a = a||b;  
    a= a&&b;  
  
}
```

16c-w.

```
function void main(){  
  
    var real c = 94.95;  
    var int a,d = 77;  
    var bool e= true;  
    e= c<d;  
    e= a<d;  
    e= c<c;  
    e= c<=d;  
    e= a<=d;  
    e= c<=c;  
    e= c>d;  
    e= a>d;  
    e= c>c;  
    e= c>=d;  
    e= a>=d;  
    e= c>=c;  
}
```

16c-nw.

```
function void main(){  
    var int a=4,b;  
    b=a&&b;  
}
```

16d-w.

```
function void main(){

    var char a = 'h';
    var real b = 94.95;
    var int c = 77;
    var char* d;
    var real* e;
    var int* f;
    var bool g= true;

    g= a==a;
    g= b==b;
    g= c==c;
    g= d==d;
    g= e==e;
    g= f==f;

    g= a!=a;
    g= b!=b;
    g= c!=c;
    g= d!=d;
    g= e!=e;
    g= f!=f;
}
```

16e-w.

```
function void main(){

    string b[6] = "abcdef";
    var int d = 77;
    d = |b|;
}
```

16f-w.

```
function void main(){  
  
    var bool e = true;  
    e = !e;  
    e = !(5<4);  
}
```

17a-nw.

```
function void main(){  
    var int y = 5;  
    var int* x;  
  
    x = &y;  
  
    x=x*5;  
}
```

17a-w.

```
function void main(){  
  
    var char x = 'h';  
    var int a = 19;  
    var char* y;  
  
    y=&x;  
    y=(y+a*17+a*5/(93+17))-(a*5);  
  
}
```

17b-nw.

```
function void main(){  
  
    var char x = 'h';  
    var char* y;  
  
    y=&x;  
    y=(y*x);  
  
}
```

18a-nw.

```
function void main(){

    var char a = 'h';
    string b[6] = "abcdef";
    var real c = 94.95;
    var int d = 77;
    var bool e = true;

    var char* ap;
    var real* cp;
    var int* dp;

    ap = &a;
    dp = &d;
    ap = &b[1];

    if ( &dp == &dp ){

    }

}
```

18a-w.

```
function void main(){

    var char a = 'h';
    string b[6] = "abcdef";
    var real c = 94.95;
    var int d = 77;
    var bool e = true;

    var char* ap;
    var real* cp;
    var int* dp;

    ap = &a;
    dp = &d;
    ap = &b[1];

}
```

19a-nw.

```
function void main(){

    var char a = 'h';
    string b[6] = "abcdef";
    var real c = 94.95;
    var int d = 77;
    var bool e = true;

    var char* ap;
    var real* cp;
    var int* dp;

    ap = &a;
    dp = &d;
    ap = &b[1];

    if (*ap == *a){
        e=false;
    }

}
```

```
function void main(){

    var char a = 'h';
    string b[6] = "abcdef";
    var real c = 94.95;
    var int d = 77;
    var bool e = true;

    var char* ap;
    var real* cp;
    var int* dp;

    ap = &a;
    dp = &d;
    ap = &b[1];

    if (*ap == a){
        e=false;
    }

}
```

Results:

=====Processing tests/10nw.t=====

ERROR: you can't to enter CHAR to INT type.

=====Processing tests/10w.t=====

-OK-

(CODE

(FUNC

f1

NONE PARAMS

(TYPE

INT

)

(BLOCK

(DECS

(VAR\_DEC

INT

(ASS\_STMT

(REGULAR\_ASS

(ID

a

)

(INT

9

)

)

)

)

(DECS

(VAR\_DEC

INT



```

                                (ASS_STMT
                                  (REGULAR_ASS
                                    (ID
                                      b
                                    )
                                    (INT
                                      7
                                    )
                                  )
                                )
                              )
                            )
                          )
                        (IF-ELSE
                          (>
                            (ID
                              a
                            )
                            (ID
                              b
                            )
                          )
                        )
                      (BLOCK
                        (RET
                          (+
                            (ID
                              b
                            )
                            (INT
                              5
                            )
                          )
                        )
                      )
                    )
                  )
                )
              )
            )
          )
        )
      )
    )
  )

```

```

    )
    )
    )
    (BLOCK
        (RET
            (*)
                (ID
                    a
                )
                (INT
                    6
                )
            )
        )
    )
    )
    )
    )
    (RET
        (INT
            0
        )
    )
    )
    )
    )
    (FUNC
        main
        NONE PARAMS
        (TYPE
            VOID
        )
        (BLOCK
            (DECS

```



```

main
NONE PARAMS
(TYPE
    VOID
)
(BLOCK
    (DECS
        (VAR_DEC
            INT
            (ASS_STMT
                (REGULAR_ASS
                    (ID
                        a
                    )
                    (INT
                        10
                    )
                )
            )
        )
    )
    (DECS
        (VAR_DEC
            BOOL
            (ASS_STMT
                (REGULAR_ASS
                    (ID
                        check
                    )
                    (BOOL
                        true
                    )
                )
            )
        )
    )
)

```

```

    )
    )
    )
    )
    )
    (IF
      (==
        (ID
          a
        )
        (INT
          10
        )
      )
    )
    (BLOCK
      (IF
        (ID
          check
        )
        (BLOCK
          (REGULAR_ASS
            (ID
              a
            )
            (INT
              2
            )
          )
        )
      )
    )
  )
)

```

```

    )
  )
)
)

=====Processing tests/12nwdowhile.t=====

DO WHILE condition has to be bool type.

=====Processing tests/12nwfor.t=====

FOR condition has to be bool type.

=====Processing tests/12nwwhile.t=====

WHILE condition has to be bool type.

=====Processing tests/12wdowhile.t=====

-OK-

(CODE
  (FUNC
    t
    NONE PARAMS
    (TYPE
      VOID
    )
    (BLOCK
      (DECS
        (VAR_DEC
          BOOL
          (ASS_STMT
            (REGULAR_ASS
              (ID
                t
              )
              (BOOL
                true
              )
            )
          )
        )
      )
    )
  )
)

```

```

        )
    )
)
(DECS
    (VAR_DEC
        INT
        (ASS_STMT
            (REGULAR_ASS
                (ID
                    c
                )
                (INT
                    10
                )
            )
        )
    )
)
)
)
(DECS
    (VAR_DEC
        INT
        (ASS_STMT
            (REGULAR_ASS
                (ID
                    d
                )
                (INT
                    100
                )
            )
        )
    )
)
)
)

```

```

        )
    )
)
(DO_WHILE
    (BLOCK
        (REGULAR_ASS
            (ID
                d
            )
            (-
                (ID
                    d
                )
                (INT
                    10
                )
            )
        )
    )
)
(IF
    (>
        (ID
            c
        )
        (ID
            d
        )
    )
    (BLOCK
        (REGULAR_ASS
            (ID
                t
            )

```





```

(CODE
  (FUNC
    t
    NONE PARAMS
    (TYPE
      VOID
    )
    (BLOCK
      (DECS
        (VAR_DEC
          BOOL
          (ASS_STMT
            (REGULAR_ASS
              (ID
                p
              )
              (BOOL
                true
              )
            )
          )
        )
      )
    )
    (DECS
      (VAR_DEC
        INT
        (ASS_STMT
          (REGULAR_ASS
            (ID
              a
            )
            (INT

```

```
0
)
)
)
)
)
(FOR
    (REGULAR_ASS
        (ID
            a
        )
        (INT
            10
        )
    )
    (ID
        p
    )
    (REGULAR_ASS
        (ID
            a
        )
        (+
            (ID
                a
            )
            (INT
                1
```



```

(FUNC
    main
    NONE PARAMS
    (TYPE
        VOID
    )
    (BLOCK
        (FUNC_CALL
            (FUNC_CALL
                t
            )
        )
    )
)
)

=====Processing tests/12wwhile.t=====
-OK-
(CODE
    (FUNC
        t
        NONE PARAMS
        (TYPE
            VOID
        )
        (BLOCK
            (DECS
                (VAR_DEC
                    BOOL
                    (ASS_STMT
                        (REGULAR_ASS
                            (ID

```



8

```
(
    )
)
)
)
)
( WHILE
    ( ID
        p
    )
    ( BLOCK
        ( IF
            (<
                ( ID
                    a
                )
                ( ID
                    b
                )
            )
        )
        ( BLOCK
            ( REGULAR_ASS
                ( ID
                    p
                )
            )
            ( BOOL
                false
            )
        )
    )
)
```





```

)
(BLOCK
  (DECS
    (STR_DEC
      (=
        str1
        (INT
          100
        )
        (STRING
          "TestingIndexType"
        )
      )
    )
  )
  (DECS
    (VAR_DEC
      INT
      (ASS_STMT
        (REGULAR_ASS
          (ID
            k
          )
          (INT
            1
          )
        )
      )
    )
  )
  (DECS
    (VAR_DEC
      BOOL

```

```

                                (ASS_STMT
                                  (REGULAR_ASS
                                    (ID
                                      t
                                    )
                                    (BOOL
                                      true
                                    )
                                  )
                                )
                              )
                            )
                          )
                        )
                      )
                    )
                  )
                )
              )
            )
          )
        )
      )
    )
  )
(WHILE
  (ID
    t
  )
  (BLOCK
    (REGULAR_ASS
      (ID
        k
      )
      (*
        (ID
          k
        )
        (INT
          2
        )
      )
    )
  )
)

```

```

)
(STR_ASS
    (ID
        str1
    )
    (ID
        k
    )
    (CHAR
        'n'
    )
)
)
(IF
    (<
        (ID
            k
        )
        (*
            (INT
                6
            )
            (INT
                2
            )
        )
    )
)
(BLOCK
    (REGULAR_ASS
        (ID
            t
        )
    )
)

```



```

                                str2
                                (INT
                                    50
                                )
                                (STRING
                                    "OperatorTest"
                                )
                            )
                        )
                    )
                (STR_ASS
                    (ID
                        str2
                    )
                    (/
                        (ID
                            b
                        )
                        (INT
                            2
                        )
                    )
                )
                (CHAR
                    'N'
                )
            )
        )
    (RET
        (INT
            1
        )
    )
)

```

```

    )
)
(FUNC
    main
    NONE PARAMS
    (TYPE
        VOID
    )
    (BLOCK
        (DECS
            (VAR_DEC
                INT
                (ASS_STMT
                    (REGULAR_ASS
                        (ID
                            v1
                        )
                        (INT
                            5
                        )
                    )
                )
            )
        )
        (DECS
            (VAR_DEC
                INT
                (ASS_STMT
                    (REGULAR_ASS
                        (ID
                            v2
                        )

```



```

                                (ID
                                  a
                                )
                                (INT
                                  4
                                )
                              )
        )
    )
  (DECS
    (VAR_DEC
      INT
      (ASS_STMT
        (REGULAR_ASS
          (ID
            b
          )
          (INT
            7
          )
        )
      )
    )
  )
  (DECS
    (VAR_DEC
      INT
      (ASS_STMT
        (REGULAR_ASS
          (ID
            c
          )

```



```
(+
    (ID
        a
    )
    (ID
        b
    )
)
)
)
)
(DECS
    (VAR_DEC
        REAL
        (ASS_STMT
            (REGULAR_ASS
                (ID
                    d
                )
                (REAL
                    2.2
                )
            )
        )
    )
)
)
)
(DECS
    (VAR_DEC
        REAL
        (ASS_STMT
            (REGULAR_ASS
                (ID
```

```

e
)
  (REAL
    3.1
  )
)
)
)
  (DECS
    (VAR_DEC
      REAL
      (ASS_STMT
        (REGULAR_ASS
          (ID
            f
          )
          (ID
            e
          )
        )
      )
    )
  )
  (DECS
    (VAR_DEC
      BOOL
      (ASS_STMT
        (REGULAR_ASS
          (ID

```

```

        c1
    )

    (BOOL
        true
    )

    )

    )

    )
    (DECS
        (VAR_DEC

BOOL

(ASS_STMT

(REGULAR_ASS

    (ID

        c2

    )

    (ID

        c1

    )

)

)
    )

```

)

(DECS

(STR\_DEC

 $(=$ 

str1

(INT

10

)

(STRING

"Testing.."

)

)

)

)

)

)

)

)

)

)

)

)

(STR\_ASS

(ID

=====Processing tests/15w2.t=====

-OK-

(CODE

(FUNC

main

NONE PARAMS

(TYPE

VOID

)

(BLOCK

(DECS

(VAR\_DEC

P\_CHAR

(ASS\_STMT

(REGULAR\_ASS

(ID

p

)

NULL



```

    )
)
=====Processing tests/16a-nw.t=====
ERROR, invalid ! to INT
=====Processing tests/16a-w.t=====
-OK-
(CODE
  (FUNC
    main
    NONE PARAMS
    (TYPE
      VOID
    )
    (BLOCK
      (DECS
        (VAR_DEC
          REAL
          (ASS_STMT
            (REGULAR_ASS
              (ID
                c
              )
              (REAL
                94.95
              )
            )
          )
        )
      )
    )
  )
  (DECS
    (VAR_DEC
      INT

```

```

                                (ASS_STMT
                                  (REGULAR_ASS
                                    (ID
                                      d
                                    )
                                    (INT
                                      77
                                    )
                                  )
                                )
                                (VAR
                                  e
                                )
                              )
                            )
                          )
                        )
                      (REGULAR_ASS
                        (ID
                          e
                        )
                        (+
                          (ID
                            d
                          )
                          (ID
                            e
                          )
                        )
                      )
                    )
                  (REGULAR_ASS
                    (ID

```



```

                                c
                            )
                        (+
                            (ID
                                c
                            )
                        (ID
                            d
                        )
                    )
                )
            (REGULAR_ASS
                (ID
                    c
                )
            (+
                (ID
                    c
                )
            (ID
                c
            )
        )
    )
)

```

=====Processing tests/16b-nw.t=====

ERROR, invalid |INT|(STR LEN) work only on string.

=====Processing tests/16b-w.t=====

-OK-

```

(CODE
  (FUNC
    main
    NONE PARAMS
    (TYPE
      VOID
    )
    (BLOCK
      (DECS
        (VAR_DEC
          BOOL
          (ASS_STMT
            (REGULAR_ASS
              (ID
                a
              )
              (BOOL
                true
              )
            )
          )
          (ASS_STMT
            (REGULAR_ASS
              (ID
                b
              )
              (BOOL
                false
              )
            )
          )
        )
      )
    )
  )
)

```



=====Processing tests/16c-w.t=====

-OK-

(CODE

(FUNC

main

NONE PARAMS

(TYPE

VOID

)

(BLOCK

(DECS

(VAR\_DEC

REAL

(ASS\_STMT

(REGULAR\_ASS

(ID

c

)

(REAL

94.95

)

)

)

)

(DECS

(VAR\_DEC

INT

(VAR

a

(ASS\_STMT

(REGULAR\_ASS

```

                                (ID
                                    d
                                )
                                (INT
                                    77
                                )
                            )
                        )
                    )
                )
            (DECS
                (VAR_DEC
                    BOOL
                    (ASS_STMT
                        (REGULAR_ASS
                            (ID
                                e
                            )
                            (BOOL
                                true
                            )
                        )
                    )
                )
            )
        )
    )
    (REGULAR_ASS
        (ID
            e
        )
    )

```

```

        (<
            (ID
                c
            )
            (ID
                d
            )
        )
    )
(REGULAR_ASS
    (ID
        e
    )
    (<
        (ID
            a
        )
        (ID
            d
        )
    )
)
(REGULAR_ASS
    (ID
        e
    )
    (<
        (ID
            c
        )
        (ID

```

```

                                c
                                )
                            )
                    )
(REGULAR_ASS
    (ID
        e
    )
    (<=
        (ID
            c
        )
        (ID
            d
        )
    )
)
(REGULAR_ASS
    (ID
        e
    )
    (<=
        (ID
            a
        )
        (ID
            d
        )
    )
)
(REGULAR_ASS

```

```

        (ID
            e
        )
        (<=
            (ID
                c
            )
            (ID
                c
            )
        )
    )
)
(REGULAR_ASS
    (ID
        e
    )
    (>
        (ID
            c
        )
        (ID
            d
        )
    )
)
)
(REGULAR_ASS
    (ID
        e
    )
    (>
        (ID

```



```

                                a
                                )
                                (ID
                                d
                                )
                                )
                                )
                                )
                                (REGULAR_ASS
                                (ID
                                e
                                )
                                (>
                                (ID
                                c
                                )
                                (ID
                                c
                                )
                                )
                                )
                                )
                                (REGULAR_ASS
                                (ID
                                e
                                )
                                (>=
                                (ID
                                c
                                )
                                (ID
                                d
                                )

```



=====Processing tests/16c-nw.t=====

ERROR, invalid &&(LOGIC AND) to INT and INT

=====Processing tests/16d-w.t=====

-OK-

(CODE

(FUNC

main

NONE PARAMS

(TYPE

VOID

)

(BLOCK

(DECS

(VAR\_DEC

CHAR

(ASS\_STMT

(REGULAR\_ASS

(ID

a

)

(CHAR

'h'

)

)

)

)

(DECS

(VAR\_DEC

REAL

(ASS\_STMT

(REGULAR\_ASS

```

                                (ID
                                    b
                                )
                                (REAL
                                    94.95
                                )
                            )
                        )
                    )
                )
            (DECS
                (VAR_DEC
                    INT
                    (ASS_STMT
                        (REGULAR_ASS
                            (ID
                                c
                            )
                            (INT
                                77
                            )
                        )
                    )
                )
            )
        (DECS
            (VAR_DEC
                P_CHAR
                (VAR
                    d
                )
            )
        )
    (DECS

```

```

(VAR_DEC
    P_REAL
    (VAR
        e
    )
)
(DECS
    (VAR_DEC
        P_INT
        (VAR
            f
        )
    )
    (DECS
        (VAR_DEC
            BOOL
            (ASS_STMT

```

```

(REGULAR_ASS

```

```

(ID

```

```

    g

```

```

)

```

```

(BOOL

```

```

    true

```

```

)

```

```

)

```

```

)

```



```

        )
    )
)
(REGULAR_ASS
    (ID
        g
    )
    (==
        (ID
            c
        )
        (ID
            c
        )
    )
)
)
(REGULAR_ASS
    (ID
        g
    )
    (==
        (ID
            d
        )
        (ID
            d
        )
    )
)
)
(REGULAR_ASS
    (ID

```

```

        g
    )
    (==
        (ID
            e
        )
        (ID
            e
        )
    )
)
(REGULAR_ASS
    (ID
        g
    )
    (==
        (ID
            f
        )
        (ID
            f
        )
    )
)
)
(REGULAR_ASS
    (ID
        g
    )
    (!=
        (ID
            a

```



```

        )
        (ID
            a
        )
    )
)
(REGULAR_ASS
    (ID
        g
    )
    (!=
        (ID
            b
        )
        (ID
            b
        )
    )
)
)
(REGULAR_ASS
    (ID
        g
    )
    (!=
        (ID
            c
        )
        (ID
            c
        )
    )
)

```

```

)
(REGULAR_ASS
    (ID
        g
    )
    (!=
        (ID
            d
        )
        (ID
            d
        )
    )
)
)
(REGULAR_ASS
    (ID
        g
    )
    (!=
        (ID
            e
        )
        (ID
            e
        )
    )
)
)
(REGULAR_ASS
    (ID
        g
    )

```

```

                (!=
                    (ID
                        f
                    )
                (ID
                    f
                )
            )
        )
    )
)

=====Processing tests/16e-w.t=====
-OK-
(CODE
    (FUNC
        main
        NONE PARAMS
        (TYPE
            VOID
        )
        (BLOCK
            (DECS
                (STR_DEC
                    (=
                        b
                    (INT
                        6
                    )
                (STRING
                    "abcdef"

```



)

=====Processing tests/16f-w.t=====

-OK-

(CODE

(FUNC

main

NONE PARAMS

(TYPE

VOID

)

(BLOCK

(DECS

(VAR\_DEC

BOOL

(ASS\_STMT

(REGULAR\_ASS

(ID

e

)

(BOOL

true

)

)

)

)

)

(REGULAR\_ASS

(ID

e

)

(!



```

(FUNC
  main
  NONE PARAMS
  (TYPE
    VOID
  )
  (BLOCK
    (DECS
      (VAR_DEC
        CHAR
        (ASS_STMT
          (REGULAR_ASS
            (ID
              x
            )
            (CHAR
              'h'
            )
          )
        )
      )
    )
  )
  (DECS
    (VAR_DEC
      INT
      (ASS_STMT
        (REGULAR_ASS
          (ID
            a
          )
          (INT

```

```

)
)
)
)
(DECS
(VAR_DEC
P_CHAR
(VAR
y
)
)
)
)
)
)
(REGULAR_ASS
(ID
y
)
(ADDRESS
(ID
x
)
)
)
)
(REGULAR_ASS
(ID
y
)
(-
((
(+

```



```

(+
  (ID
    y
  )
  (*
    (ID
      a
    )
    (INT
      17
    )
  )
)
(/
  (*
    (ID
      a
    )
    (INT
      5
    )
  )
  ((
    (+
      (INT
        93
      )
      (INT
        17
      )
    )
  )
)

```



```

NONE PARAMS
(TYPE
    VOID
)
(BLOCK
    (DECS
        (VAR_DEC
            CHAR
            (ASS_STMT
                (REGULAR_ASS
                    (ID
                        a
                    )
                    (CHAR
                        'h'
                    )
                )
            )
        )
    )
    (DECS
        (STR_DEC
            (=
                b
                (INT
                    6
                )
                (STRING
                    "abcdef"
                )
            )
        )
    )
)

```

```

(DECS
  (VAR_DEC
    REAL
    (ASS_STMT
      (REGULAR_ASS
        (ID
          c
        )
        (REAL
          94.95
        )
      )
    )
  )
)
(DECS
  (VAR_DEC
    INT
    (ASS_STMT
      (REGULAR_ASS
        (ID
          d
        )
        (INT
          77
        )
      )
    )
  )
)
(DECS
  (VAR_DEC
    BOOL

```



dp

```
(REGULAR_ASS
    (ID
        ap
    )
    (ADDRESS
        (ID
            a
        )
    )
)
(REGULAR_ASS
    (ID
        dp
    )
    (ADDRESS
        (ID
            d
        )
    )
)
```

```

        )
        (REGULAR_ASS
            (ID
                ap
            )
        (ADDRESS
            (ID
                b
            )
            (IN_PLACE
                (INT
                    1
                )
            )
        )
    )
)

=====Processing tests/19a-nw.t=====
ERROR, invalid *(pointer) to CHAR
=====Processing tests/19a-w.t=====
-OK-
(CODE
    (FUNC
        main
        NONE PARAMS
        (TYPE
            VOID
        )
        (BLOCK

```

```

(DECS
  (VAR_DEC
    CHAR
    (ASS_STMT
      (REGULAR_ASS
        (ID
          a
        )
        (CHAR
          'h'
        )
      )
    )
  )
)
(DECS
  (STR_DEC
    (=
      b
      (INT
        6
      )
      (STRING
        "abcdef"
      )
    )
  )
)
(DECS
  (VAR_DEC
    REAL
    (ASS_STMT
      (REGULAR_ASS

```



```

                                (ID
                                    c
                                )
                                (REAL
                                    94.95
                                )
                            )
                        )
                    )
                (DECS
                    (VAR_DEC
                        INT
                        (ASS_STMT
                            (REGULAR_ASS
                                (ID
                                    d
                                )
                                (INT
                                    77
                                )
                            )
                        )
                    )
                )
            (DECS
                (VAR_DEC
                    BOOL
                    (ASS_STMT
                        (REGULAR_ASS
                            (ID
                                e
                            )
                        )
                    )
                )
            )
        )
    )

```

```

                                (BOOL
                                true
                                )
                            )
                        )
                    )
                )
            (DECS
                (VAR_DEC
                    P_CHAR
                    (VAR
                        ap
                    )
                )
            )
            (DECS
                (VAR_DEC
                    P_REAL
                    (VAR
                        cp
                    )
                )
            )
            (DECS
                (VAR_DEC
                    P_INT
                )
            )
            (VAR
                dp
            )
        )
    )
)

```



```

        (ID
          b
        )
      (IN_PLACE
        (INT
          1
        )
      )
    )
  )
  (IF
    (==
      (PTR
        (ID
          ap
        )
      )
      (ID
        a
      )
    )
    (BLOCK
      (REGULAR_ASS
        (ID
          e
        )
        (BOOL
          false
        )
      )
    )
  )
)

```

```

        )
    )
)

=====Processing tests/1a-nw.t=====

ERROR : no main func.

=====Processing tests/1a-w.t=====

-OK-

(CODE
    (FUNC
        foo
        NONE PARAMS
        (TYPE
            INT
        )
        (BLOCK
            (DECS
                (VAR_DEC
                    INT
                    (VAR
                        x
                    )
                )
            )
        )
    )
    (BLOCK
        (DECS
            (VAR_DEC
                INT
                (VAR
                    y
                )
            )
        )
    )
)

```

```

        )
    )
    (REGULAR_ASS
        (ID
            x
        )
        (INT
            1
        )
    )
)
(REGULAR_ASS
    (ID
        y
    )
    (INT
        2
    )
)
)
(BLOCK
    (REGULAR_ASS
        (ID
            x
        )
        (INT
            2
        )
    )
)
)
(REGULAR_ASS
    (ID
        y

```

```

    )
    (INT
        3
    )
    )
    )
    (RET
        (INT
            0
        )
    )
    )
    )
    )
    (FUNC
        main
        NONE PARAMS
        (TYPE
            VOID
        )
        (BLOCK
            (BLOCK
                BLOCK
            )
        )
    )
    )
    )
    )

```

=====Processing tests/1b-nw.t=====

Main function have to be at global scope

=====Processing tests/1b-w.t=====

-OK-

(CODE

```

(FUNC
    main
    NONE PARAMS
    (TYPE
        VOID
    )
    (BLOCK
        (BLOCK
            BLOCK
        )
    )
)
(FUNC
    foo
    NONE PARAMS
    (TYPE
        INT
    )
    (BLOCK
        (DECS
            (VAR_DEC
                INT
                (VAR
                    x
                )
            )
        )
    )
    (BLOCK
        (DECS
            (VAR_DEC
                INT
            )
        )
    )
)

```



```

                (VAR
                    y
                )
            )
        )
    (REGULAR_ASS
        (ID
            x
        )
        (INT
            1
        )
    )
    (REGULAR_ASS
        (ID
            y
        )
        (INT
            2
        )
    )
    (BLOCK
        (REGULAR_ASS
            (ID
                x
            )
            (INT
                2
            )
        )
    )
)

```



```

    )
  (BLOCK
    (RET
      (INT
        0
      )
    )
  )
)
)
(FUNC
  foo_2
  NONE PARAMS
  (TYPE
    INT
  )
  (BLOCK
    (DECS
      (VAR_DEC
        INT
        (VAR
          a
        )
      )
    )
  )
  (REGULAR_ASS
    (ID
      a
    )
    (INT
      2
    )
  )
)

```

```

        )
      (RET
        (ID
          a
        )
      )
    )
  )
)
(FUNC
  foo_3
  NONE PARAMS
  (TYPE
    INT
  )
  (BLOCK
    (IF
      (BOOL
        true
      )
      (BLOCK
        (RET
          (FUNC_CALL
            foo
          )
        )
      )
    )
  )
  (RET
    (INT
      0
    )
  )
)

```



=====Processing tests/3-nw.t=====

ERROR, func foo\_2 already exist in scope

=====Processing tests/3-w.t=====

-OK-

(CODE

  (FUNC

    foo

    NONE PARAMS

    (TYPE

      INT

    )

    (BLOCK

      (RET

        (INT

          0

        )

      )

    )

  )

  (FUNC

    foo\_2

    NONE PARAMS

    (TYPE

      INT

    )

    (BLOCK

      (DECS

        (VAR\_DEC

          INT

        (VAR

          a

```

        )
    )
)
(REGULAR_ASS
    (ID
        a
    )
    (INT
        2
    )
)
)
(RET
    (ID
        a
    )
)
)
)
(FUNC
    foo_3
    NONE PARAMS
    (TYPE
        INT
    )
    (BLOCK
        (DECS
            (FUNC_DEC
                (FUNC
                    foo_2
                    NONE PARAMS
                    (TYPE

```

```

                                INT
                                )
                                (BLOCK
                                (DECS
                                (VAR_DEC
                                INT
                                (VAR
                                a
                                )
                                )
                                )
                                )
                                (REGULAR_ASS
                                (ID
                                a
                                )
                                (INT
                                2
                                )
                                )
                                (RET
                                (ID
                                a
                                )
                                )
                                )
                                )
                                )
                                )
                                (IF
                                (BOOL
                                true

```





```

        (REGULAR_ASS
            (ID
                a
            )
            (INT
                2
            )
        )
    )
)

```

=====Processing tests/4a-nw.t=====

ERROR: identifier x already exist in current function.

=====Processing tests/4a-w.t=====

-OK-

```

(CODE
    (FUNC
        foo
        NONE PARAMS
        (TYPE
            INT
        )
        (BLOCK
            (DECS
                (VAR_DEC
                    INT
                    (VAR
                        x
                    )
                )
            )
            (DECS

```

```

(VAR_DEC
    P_INT
    (VAR
        I
    )
)
(DECS
    (VAR_DEC
        P_CHAR
        (VAR
            k
        )
    )
    (DECS
        (STR_DEC
            (STRING
                y
                (INT
                    10
                )
            )
        )
        (DECS
            (VAR_DEC
                CHAR
                (VAR
                    z
                )
            )
        )
    )
)

```

```

        )
    )
)
(REGULAR_ASS
    (ID
        x
    )
    (INT
        5
    )
)
)
(REGULAR_ASS
    (ID
        y
    )
    (STRING
        "foobar"
    )
)
)
(REGULAR_ASS
    (ID
        k
    )
    (ADDRESS
        (ID
            y
        )
        (IN_PLACE
            (INT
                5
            )
        )
    )
)

```

```

        )
    )
)
(REGULAR_ASS
    (ID
        z
    )
    (PTR
        (ID
            k
        )
        (INT
            5
        )
    )
)
)
)
(REGULAR_ASS
    (ID
        y
    )
    (STRING
        "barfoo"
    )
)
)
(RET
    (PTR
        (ID
            l
        )
    )
)
)
)

```

```

        )
    )
    (FUNC
        main
        NONE PARAMS
        (TYPE
            VOID
        )
        BLOCK
    )
)

=====Processing tests/4b-nw.t=====
ERROR: identifier x already exist in current function.

=====Processing tests/5a-nw.t=====
ERROR: at function: foo20, function: foo43 didnt found.
you need to declare on function before you can use her.

=====Processing tests/5b-nw.t=====
ERROR: at function: foo1, function: foo2 didnt found.
you need to declare on function before you can use her.

=====Processing tests/5-w.t=====
-OK-

(CODE
    (FUNC
        foo1
        (PARAMS
            INT
            a b y
            CHAR
            c
        )
        (TYPE

```

```

        BOOL
    )
(BLOCK
    (DECS
        (VAR_DEC
            REAL
            (VAR
                s
            )
        )
    )
    (DECS
        (VAR_DEC
            P_INT
            (ASS_STMT
                (REGULAR_ASS
                    (ID
                        k
                    )
                    NULL
                )
                (VAR
                    m
                )
            )
        )
    )
    (DECS
        (VAR_DEC
            INT
            (VAR
                g
            )
            (VAR

```

```

)
)
)
(DECS
  (VAR_DEC
    STRING
    (VAR
      p
    )
  )
)
(DECS
  (VAR_DEC
    BOOL
    (ASS_STMT
      (REGULAR_ASS
        (ID
          res
        )
        (BOOL
          true
        )
      )
    )
  )
)
)
)
(DECS
  (VAR_DEC
    P_CHAR
    (VAR
      f
    )
  )
)

```



```
(REGULAR_ASS
    (ID
        s
    )
    (REAL
        4.5
    )
)
(REGULAR_ASS
    (ID
        c
    )
    (PTR
        (ID
            f
        )
    )
)
)
(REGULAR_ASS
    (ID
        i
```

```

    )
    (-
        (ID
            f
        )
        (INT
            7
        )
    )
)
(REGULAR_ASS
    (ID
        c
    )
    (ID
        c
    )
)
(BLOCK
    (DECS
        (VAR_DEC
            CHAR
            (ASS_STMT
                (REGULAR_ASS
                    (ID
                        x
                    )
                    (CHAR
                        'd'
                    )
                )
            )
        )
    )
)

```

```

                                (VAR
                                    b
                                )
                            )
                )
        (DECS
            (VAR_DEC
                INT
                (VAR
                    y
                )
            )
        )
    )
    (REGULAR_ASS
        (ID
            b
        )
        (CHAR
            '&'
        )
    )
    (REGULAR_ASS
        (ID
            a
        )
        (-
            (/
                ((
                    (*
                        (ID

```

```

                                y
                                )
                                (INT
                                7
                                )
                                )
                                )
                                )
                                (ID
                                a
                                )
                                )
                                (ID
                                y
                                )
                                )
                                )
                                (REGULAR_ASS
                                (ID
                                res
                                )
                                (&&
                                ((
                                (==
                                (ID
                                b
                                )
                                (ID
                                c
                                )
                                )
                                )

```

```
(TYPE VOID  
    )  
fee1  
(PARAMS INT i j k x  
    )  
(FUNC  
    )  
    )  
        )  
            res  
                (ID  
                    (RET  
                        )  
                            )  
                                )  
                                    ((  
                                        )  
                                            )
```

```

)
(BLOCK
    (DECS
        (FUNC_DEC
            (FUNC
                fee2
                (PARAMS
                    INT
                    l m n
                )
                (TYPE
                    BOOL
                )
                (BLOCK
                    (DECS
                        (VAR_DEC
                            BOOL
                            (VAR
                                x
                                (VAR
                                    j
                                )
                            )
                        )
                    )
                )
            )
            (DECS
                (VAR_DEC
                    P_CHAR
                    (VAR
                        k
                    )
                )
            )
        )
    )
)

```

```

        )
    )
    (PTR_ASS
        (ID
            k
        )
        (CHAR
            '@'
        )
    )
)
(REGULAR_ASS
    (ID
        i
    )
    (+
        (ID
            l
        )
        (ID
            l
        )
    )
)
)
(IF
    (||
        ((
            (==
                (PTR
                    (ID

```

k





(ID

m

)

)

(ID

i

)

)

)

)

)

)

(BLOCK

(REGULAR\_ASS

(ID

x

)

(<

(ID

I

)

(ID

m

)

)

)

)

)

```

      (RET
      (ID
      x
      )
      )
      )
      )
      )
      )
      (DECS
      (VAR_DEC
      BOOL
      (VAR
      ghgh
      )
      )
      )
      )
      )
      )
      (BLOCK
      (DECS
      (VAR_DEC
      CHAR
      (VAR
      x
      )
      )
      )
      (DECS
      (VAR_DEC
      BOOL
      (ASS_STMT
      (REGULAR_ASS
      (ID

```

```
(REGULAR_ASSIGNMENT
  (ID k)
  (FUNC_CALL foo1
    (ARGS
      (INT 5)
      (ARGS
        (ID i)
        (ARGS
          (INT 34)
          (ID x)
        )
      )
    )
  )
)
```

[illegible]



```

        )
    )
    (VAR
        cnt
    )
)
)
)
(DECS
    (STR_DEC
        (STRING
            a
            (INT
                30
            )
            (=
                b
                (INT
                    100
                )
                (ID
                    s3
                )
            )
        )
    )
)
)
)
(DECS
    (VAR_DEC
        CHAR
        (VAR
            c

```

```
(ID ds  
)  
(REGULAR_ASS  
(CHAR 'e'  
)  
(STR_ASS  
)  
(ID c  
)  
(REGULAR_ASS  
)  
(ID  
(REGULAR_ASS  
)  
(ID i  
)  
(ASS_STMT  
INT  
(VAR_DEC  
(DECS  
)  
)
```

```

        a
    )
    (INT
        19
    )
    (CHAR
        'f'
    )
)
(STR_ASS
    (ID
        a
    )
    (+
        (INT
            4
        )
        (INT
            2
        )
    )
    (CHAR
        'g'
    )
)
(REGULAR_ASS
    (ID
        b
    )
    (ID
        a
    )
)

```



```

        )
    )
    (STR_ASS
        (ID
            b
        )
        (INT
            3
        )
        (ID
            c
        )
    )
)
(REGULAR_ASS
    (ID
        a
    )
    (STRING
        "test"
    )
)
(REGULAR_ASS
    (ID
        i
    )
    (STR_LEN
        (ID
            b
        )
    )
)
)

```

```

(REGULAR_ASS
    (ID
        cnt
    )
    (INT
        1
    )
)
(FOR
    (REGULAR_ASS
        (ID
            i
        )
        (INT
            0
        )
    )
    (<
        (ID
            i
        )
        (STR_LEN
            (ID
                s1
            )
        )
    )
    (REGULAR_ASS
        (ID
            i
        )

```

```

        (+
          (ID
            i
          )
          (INT
            1
          )
        )
      )
    )
  (BLOCK
    (DO_WHILE
      (BLOCK
        (IF
          (==
            (ID
              (ID
                s1
              )
              (IN_PLACE
                (ID
                  i
                )
              )
            )
            (ID
              (ID
                s2
              )
              (IN_PLACE
                (ID
                  j

```

```

    )
    )
    )
    )
    (REGULAR_ASS
        (ID
            cnt
        )
        (*
            (ID
                cnt
            )
            (INT
                2
            )
        )
    )
    )
    )
    (REGULAR_ASS
        (ID
            j
        )
        (+
            (ID
                j
            )
            (INT
                1
            )
        )
    )
    )
    )

```



```

)
(FUNC
    main
    NONE PARAMS
    (TYPE
        VOID
    )
    (BLOCK
        (DECS
            (VAR_DEC
                INT
                (VAR
                    a
                )
            )
        )
    )
    (REGULAR_ASS
        (ID
            a
        )
        (FUNC_CALL
            foo2
        )
    )
)
)
(FUNC
    foo4
    NONE PARAMS
    (TYPE
        INT

```

```

)
(BLOCK
  (DO_WHILE
    (BLOCK
      (DECS
        (VAR_DEC
          INT
          (VAR
            j
          )
        )
      )
    )
  )
  (REGULAR_ASS
    (ID
      j
    )
    (+
      (ID
        j
      )
      (INT
        1
      )
    )
  )
)
)
)
(BLOCK
  (BOOL
    true
  )
)
)
(BLOCK

```

```

(DECS
  (VAR_DEC
    INT
    (VAR
      x
      (VAR
        k
      )
    )
  )
)
(DECS
  (VAR_DEC
    P_INT
    (VAR
      y
    )
  )
)
)
(REGULAR_ASS
  (ID
    x
  )
  (INT
    5
  )
)
)
(REGULAR_ASS
  (ID
    y
  )
)

```



```

        (ADDRESS
            (ID
                x
            )
        )
    )
)
(REGULAR_ASS
    (ID
        x
    )
    (INT
        6
    )
)
)
(IF
    (&&
        (==
            (ADDRESS
                (ID
                    x
                )
            )
            (ID
                y
            )
        )
        (==
            (PTR
                (ID
                    y
                )
            )
        )
    )
)

```

```

        )
        (ID
            x
        )
    )
)
(PTR_ASS
    (ID
        y
    )
    (INT
        9
    )
)
)
)
(BLOCK
    (DECS
        (VAR_DEC
            P_CHAR
            (VAR
                x
            )
        )
        (DECS
            (STR_DEC
                (STRING
                    y
                )
                (INT
                    10
                )
            )
        )
    )
)

```



```
(REGULAR_ASS
    (ID
        z
    )
    (PTR
        (ID
            x
        )
        (INT
            5
        )
    )
)
```

```

                                (REGULAR_ASS
                                    (ID
                                        y
                                    )
                                (STRING
                                    "barfoo"
                                )
                            )
                        )
                    )
                (RET
                    (INT
                        0
                    )
                )
            )
        )
    )
(FUNC
    foo10
    NONE PARAMS
    (TYPE
        INT
    )
    (BLOCK
        (DECS
            (FUNC_DEC
                (FUNC
                    foo20
                    NONE PARAMS
                    (TYPE
                        INT

```

```

        )
        BLOCK
    )
)
(DECS
    (FUNC_DEC
        (FUNC
            foo30
            NONE PARAMS
            (TYPE
                INT
            )
            BLOCK
        )
    )
)
(DECS
    (FUNC_DEC
        (FUNC
            foo40
            NONE PARAMS
            (TYPE
                INT
            )
            (BLOCK
                (DECS
                    (FUNC_DEC
                        (FUNC

```

foo41

NONE PARAMS

(TYPE

INT

)

(BLOCK

(DECS

(VAR\_DEC

CHAR

(VAR

i

)

)

)

)

)

)

(DECS

(FUNC\_DEC

(FUNC

foo42

NONE PARAMS

(TYPE





```
(  
)  
  
)  
  
)  
  
(DECS  
  
(FUNC_DEC  
  
(FUNC  
  
foo43  
  
NONE PARAMS  
  
(TYPE  
  
VOID  
  
)  
  
(BLOCK  
  
(DECS  
  
(VAR_DEC  
  
INT  
  
(VAR  
  
i  
  
(VAR  
  
j
```

(VAR

k

(VAR

**X**

(VAR

1

(VAR

m

)

)

)

)

)

)

)

(DECS

(VAR\_DEC

BOOL

(VAR

```

df
)
)
)
)
)
(IF
(==
(FUNC_CALL
fee1
(ARGS
(ID
i
)
(ARGS
(ID
j
)
(ARGS
(ID

```

k

)

(ID

**X**

)

)

)

)

)

(FUNC\_CALL

foo43

)

)

(BLOCK

```
(REGULAR_ASS
```

(ID

df

)

( $<$

(ID

I

)

(ID

m

)

)

)

)

)

)

)

)

)

)

)

)

)

)

)

)

)

)

```

    )
)
=====Processing tests/6a-w.t=====
-OK-
(CODE
    (FUNC
        main
        NONE PARAMS
        (TYPE
            VOID
        )
        (BLOCK
            (DECS
                (VAR_DEC
                    INT
                    (VAR
                        i
                    )
                )
            )
        )
        (IF
            (>
                (INT
                    3
                )
                (INT
                    2
                )
            )
        )
        (BLOCK
            (REGULAR_ASS

```

```

                                (ID
                                    i
                                )
                                (INT
                                    5
                                )
                            )
                        )
                    )
                )
            )
        )
    )
)

=====Processing tests/6b-w.t=====

-OK-

(CODE
    (FUNC
        main
        NONE PARAMS
        (TYPE
            VOID
        )
        (BLOCK
            (IF
                (>
                    (INT
                        3
                    )
                    (INT
                        2
                    )
                )
            )
        )
    )
)

```





```

(CODE
  (FUNC
    foo
    (PARAMS
      INT
      a b c
    )
    (TYPE
      INT
    )
    (BLOCK
      (DECS
        (VAR_DEC
          INT
          (VAR
            x
          )
        )
      )
    )
    (BLOCK
      (DECS
        (VAR_DEC
          INT
          (VAR
            y
          )
        )
      )
    )
    (REGULAR_ASS
      (ID
        x

```

```

        )
        (INT
            1
        )
    )
    (REGULAR_ASS
        (ID
            y
        )
        (INT
            2
        )
    )
    (BLOCK
        (REGULAR_ASS
            (ID
                x
            )
            (INT
                2
            )
        )
    )
    (REGULAR_ASS
        (ID
            y
        )
        (INT
            3
        )
    )

```

```

    )
    (RET
        (INT
            0
        )
    )
)
)
(FUNC
    foo2
    NONE PARAMS
    (TYPE
        INT
    )
    (BLOCK
        (DECS
            (VAR_DEC
                INT
                (VAR
                    a
                    (VAR
                        b
                        (VAR
                            c
                            (VAR
                                d
                                (VAR
                                    e
                                )
                            )
                        )
                    )
                )
            )
        )
    )
)

```

```

    )
  )
)
(REGULAR_ASS
  (ID
    a
  )
  (FUNC_CALL
    foo
    (ARGS
      (ID
        a
      )
      (ARGS
        (ID
          b
        )
        (ID
          c
        )
      )
    )
  )
)
)
)
(BLOCK
  BLOCK
)
(RET
  (INT
    0

```

```

)
)
)
)
(FUNC
    main
    NONE PARAMS
    (TYPE
        VOID
    )
    BLOCK
)
)

```

=====Processing tests/8-a-nw.t=====

arg type(CHAR) didnt match to param type(INT).

=====Processing tests/8-w.t=====

-OK-

```

(CODE
    (FUNC
        foo
        (PARAMS
            INT
            a b
            CHAR
            j
        )
        (TYPE
            INT
        )
        (BLOCK
            (DECS

```

```

        (VAR_DEC
            INT
            (VAR
                x
            )
        )
    )
)
(BLOCK
    (DECS
        (VAR_DEC
            INT
            (VAR
                y
            )
        )
    )
)
(REGULAR_ASS
    (ID
        x
    )
    (INT
        1
    )
)
(REGULAR_ASS
    (ID
        y
    )
    (INT
        2
    )
)

```

```

    )
    (BLOCK
        (REGULAR_ASS
            (ID
                x
            )
            (INT
                2
            )
        )
    )
    (REGULAR_ASS
        (ID
            y
        )
        (INT
            3
        )
    )
    (RET
        (INT
            0
        )
    )
)
(FUNC
    foo2
    NONE PARAMS
    (TYPE

```

```

        INT
    )
(BLOCK
    (DECS
        (VAR_DEC
            INT
            (VAR
                a
                (VAR
                    b
                    (VAR
                        c
                    )
                )
            )
        )
    )
)
(DECS
    (VAR_DEC
        CHAR
        (VAR
            d
            (VAR
                e
            )
        )
    )
)
)
(REGULAR_ASS
    (ID
        a

```



```

    )
    (FUNC_CALL
      foo
      (ARGS
        (ID
          a
        )
        (ARGS
          (ID
            b
          )
          (ID
            e
          )
        )
      )
    )
  )
)
(BLOCK
  BLOCK
)
(RET
  (INT
    0
  )
)
)
)
)
(FUNC
  main
  NONE PARAMS

```

```

        (TYPE
            VOID
        )
    BLOCK
)
)

=====Processing tests/9nw2.t=====
ERROR: function smt can not return string
=====Processing tests/9nw.t=====
ERROR: function foo declare return INT but return CHAR.
=====Processing tests/9w.t=====

-OK-

(CODE
    (FUNC
        foo
        NONE PARAMS
        (TYPE
            INT
        )
        (BLOCK
            (DECS
                (VAR_DEC
                    INT
                    (ASS_STMT
                        (REGULAR_ASS
                            (ID
                                a
                            )
                            (INT
                                10
                            )
                        )
                    )
                )
            )
        )
    )
)

```



```

                                a
                                )
                            )
                        )
                    )
                )
            (FUNC
                main
                NONE PARAMS
                (TYPE
                    VOID
                )
                (BLOCK
                    (FUNC_CALL
                        (FUNC_CALL
                            foo
                        )
                    )
                )
            )
        )
    )

```