



Τμήμα Μηχανικών Η/Υ και Πληροφορικής
Μεταφραστές
Τελική Αναφορά

Σιδηρόπουλος Παναγιώτης 4489

Μάιος 2023

Οδηγίες εκτέλεσης του μεταφραστή:

Ο μεταφραστής της CutePy υλοποιήθηκε χρησιμοποιώντας κώδικα γραμμένο σε Java.

Εντολή για μεταγλώττιση του μεταφραστή:

```
$- javac cutePy_4489.java
```

Εντολή για τη μετάφραση ενός προγράμματος x.cpy:

```
$- java cutePy_4489 x.cpy
```

Επιπλέον μπορεί να παραχθεί ο αντίστοιχος κώδικας σε C απλά προσθέτοντας το flag «-c».

Πχ

```
$- java cutePy_4489 x.cpy -c
```

Να σημειωθεί πως η μετατροπή σε γλώσσα C λειτουργεί μόνο για απλά προγράμματα (που έχουν μόνο μια κύρια συνάρτηση και όχι εμφωλευμένες τοπικές και τα declare ορίζουν όλες τις μεταβλητές που χρησιμοποιήθηκαν) καθώς και ότι λειτουργεί «τυφλά» χωρίς να λαμβάνεται υπόψιν ο πίνακας συμβόλων. Αυτό συμβαίνει διότι η υλοποίηση της μετατροπής του κώδικα σε C έγινε αμέσως μετά την υλοποίηση του ενδιάμεσου κώδικα και πριν ανακοινωθεί πως είναι προαιρετική και δεν υπήρχε χρόνος για την προσθήκη του ελέγχου του πίνακα συμβόλων.

Τρόπος Υλοποίησης

Χρησιμοποιήθηκε το GitHub ως version control system:

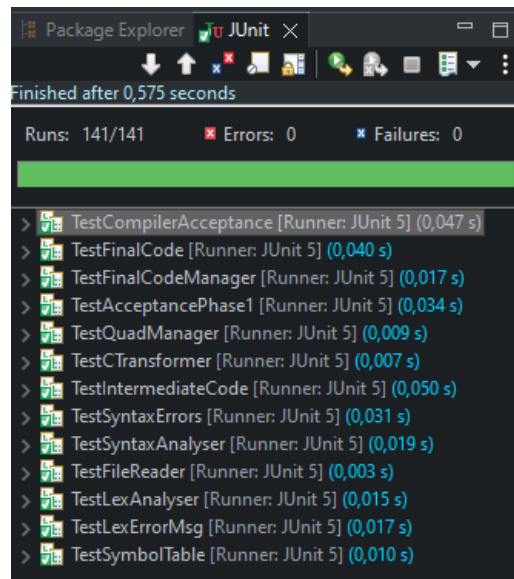
<https://github.com/PanosSid/CutePy-Compiler>

Κατά τη διάρκεια της υλοποίησης του κώδικα το repository ήταν private και έγινε public μετά την ημερομηνία παράδοσης του πρότζεκτ.

Μπορείτε να περιηγηθείτε στην ιστορία των commits για να δείτε την εξέλιξή του κώδικα στον χρόνο.

Παράλληλα με την υλοποίηση του κώδικα για τον μεταφραστή, υλοποιήθηκαν επίσης 141 test μέθοδοι χρησιμοποιώντας το testing framework JUnit 5 ώστε να μπορούν να εκτελεστούν αυτόματα αυτοί οι έλεγχοι ανά πάσα στιγμή, παρέχοντας έτσι μια δικλείδα ασφαλείας για την αποφυγή προβλημάτων κατά την υλοποίηση.

Τα τεστ αυτά βρίσκονται στον κατάλογο tests του repository, ο οποίος περιέχει τεστ για κάθε κλάση που υλοποιήθηκε. Τα τεστ για το συνολικό πρόγραμμα τα οποία ελέγχουν όλα τα παραγόμενα αρχεία (.int, .symb, .asm) βρίσκονται στην κλάση /acceptancefinal/TestCompilerAcceptance.java. Οι υπόλοιπες κλάσεις τεστάρουν μεμονωμένες λειτουργίες όπως η λεκτική ανάλυση (TestLexAnalyzer.java, TestLexErrorMsg.java), η συντακτική ανάλυση (TestSyntaxAnalyzer.java, TestSyntaxErrors.java), η παραγωγή ενδιάμεσου κώδικα (TestQuadManager.java, TestIntermediateCode.java), η παραγωγή του πίνακα συμβόλων (TestSymbolTable.java) και η παραγωγή του τελικού κώδικα (TestFinalCodeManager.java, TestFinalCode.java). Υλοποιήθηκαν και τεστ για τις συμπληρωματικές κλάσεις όπως η FileReader.java και η CTransformer.java.



Εικόνα 1 Εκτέλεση όλων των τεστ

Παρακάτω χρησιμοποιώντας το εργαλείο του Eclipse IDE μπορούμε να δούμε και το ποσοστό του κώδικα που καλύπτουν αυτά τα τεστ το οποίο είναι κοντά στο 90%!

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
▼ CutePy-Compiler				
▼ src				
▼ syntax				
> SyntaxAnalyser.java	87,2 %	1.887	276	2.163
▼ symboltable.entities				
> EntityFactory.java	64,4 %	87	48	135
> LocalFunction.java	71,3 %	102	41	143
> Variable.java	61,5 %	59	37	96
> FormalParameter.java	62,3 %	48	29	77
> Function.java	72,4 %	71	27	98
> Parameter.java	69,0 %	60	27	87
> Entity.java	53,5 %	23	20	43
> MainFunction.java	81,4 %	35	8	43
> TemporaryVariable.java	86,7 %	26	4	30
> DataType.java	100,0 %	14	0	14
> ParameterMode.java	100,0 %	31	0	31
▼ lex				
> CharTypes.java	86,9 %	456	69	525
> FileReader.java	60,9 %	95	61	156
> Token.java	54,2 %	58	49	107
> LexAnalyser.java	99,8 %	950	2	952
> EOFToken.java	100,0 %	7	0	7
▼ view				
> CutePyCompiler.java	77,4 %	202	59	261
▼ finalcode				
> FinalCodeManager.java	94,8 %	1.001	55	1.056
▼ symboltable				
> Scope.java	86,8 %	177	27	204
> SymbolTable.java	99,6 %	231	1	232
▼ intermediatecode				
> CTransformer.java	96,6 %	393	14	407
> QuadManager.java	95,2 %	178	9	187
> Quad.java	100,0 %	199	0	199
▼ exceptions				
> CutePyException.java	100,0 %	4	0	4
> tests	96,4 %	6.713	252	6.965

Έλεγχος σύνθετου προγράμματος (1):

Πρόγραμμα: primes.cpy

```
def main_primes():
#{
    #declare i

    def isPrime(x):
    #{
        #declare i

        def divides(x,y):
        #{
            if (y == (y//x) * x):
                return (1);
            else:
                return (0);
        #}

        i = 2;
        while (i<x):
        #{
            if (divides(i,x)==1):
                return (0);
            i = i + 1;
        #}
        return (1);

    #}

    # $ body of main_primes $
    i = 2;
    while (i<=30):
    #{
        if (isPrime(i)==1):
        #{
            print(i);
            i = i + 1;
        #}
        else:
            i = i + 1;
    #}
#}

if __name__ == "__main__":
    main_primes();
```

Λειτουργία:

Υπολογίζει όλους τους πρώτους αριθμούς από μικρότερους του 30 και τους τυπώνει στην κονσόλα.

Αναμενόμενο αποτέλεσμα: οι αριθμοί 2, 3, 5, 7, 11, 13, 17, 19, 23 και 29

Αποτέλεσμα εκτέλεσης:

```
1  #SIDIROPOULOS PANAGIOTIS AM:4489 cse84489
2
3  .data
4
5  str_nl: .asciz "\n"
6
7  .text
8
9  L0:
10     j main
11
12  divides:                                # 1: begin_block
13     sw ra, -0(sp)
14
15  L2:                                    # 2: //, y, x, &1
16     lw t1, -16(sp)
17     lw t2, -12(sp)
18     div t1, t1, t2
19     sw t1, -20(sp)
20
21  L3:                                    # 3: *, &1, x, &2
22     lw t1, -20(sp)
23     lw t2, -12(sp)
24     mul t1, t1, t2
25     sw t1, -24(sp)
26
27  L4:                                    # 4: ==, y, &2, 6
28     lw t1, -16(sp)
29     lw t2, -24(sp)
30     beq t1, t2, L6
31
```

Line: 1 Column: 1 ☒ Show Line Numbers

Messages Run I/O

-- program is finished running (0) --

2
3
5
7
11
13
17
19
23
29

-- program is finished running (0) --

Clear

Έλεγχος σύνθετου προγράμματος (2) – Πολύπλοκες συνθήκες:

Πρόγραμμα: gotolimits.cpy

```
def main_gotolimits():
#{
    #declare x, r
    #declare upper_limit, lower_limit

    def goToLower():
    #{
        while( x > lower_limit):
            x = x - 1;
        return(x);
    #{

    def goToUpper():
    #{
        while( x < upper_limit): #{
            x = x + 1;
        }
        return(x);
    #{

    lower_limit = int(input());
    x = int(input());
    upper_limit = int(input());

    if ([x >= 0] and [x <=0]) or
        [[lower_limit < 0 or upper_limit < 0 ]
         or [lower_limit >= upper_limit]
         or [x < lower_limit] or [x > upper_limit]] ):
        print(-99999);
    else:
    #{
        if (not [(x - lower_limit) != (upper_limit - x)]):
            print(0);
        else:
        #{
            if ((x - lower_limit) < (upper_limit - x)):
                r = goToLower();
            else:#{
                r = goToUpper();
            }
            print(r);
        }
    }

    #}
#}

if __name__ == "__main__":
    main_gotolimits();
```

Λειτουργία:

1. Δέχεται ως είσοδο από τον χρήστη
 - a. Το χαμηλότερο όριο
 - b. Έναν αριθμό (έστω x)
 - c. Το υψηλότερο όριο
2. Ελέγχει αν τα όρια και η τιμή του x είναι σωστή αλλιώς τυπώνει -99999 και τερματίζει
 - a. Το x πρέπει να είναι διαφορετικό του μηδέν
 - b. Τα όρια πρέπει να είναι μεγαλύτερα του μηδέν
 - c. Το μικρότερο όριο πρέπει να είναι μικρότερο από το μεγαλύτερο
 - d. Το x πρέπει να βρίσκεται ανάμεσα στα όρια
 - e. Αν δεν ισχύει κάτι από αυτά τυπώνει -999 και τερματίζει

3. Τυπώνει το όριο που βρίσκεται πιο κοντά στον αριθμό x καλώντας την κατάλληλη συνάρτηση η οποία αυξάνει/μειώνει τον x μέχρι να φτάσει στο κοντινότερο όριο
- a. Εάν το x ισαπέχει από τα όρια τυπώνει 0

Αναμενόμενο αποτέλεσμα:

	Test1	Test2	Test3	Test4	Test5	Test6	Test7
lower_limit	5	5	5	5	5	5	10
x	0	12	2	10	8	6	11
upper_limit	10	10	10	15	10	10	3
r	-99999	-99999	-99999	0	10	5	-99999

Αποτέλεσμα εκτέλεσης:

EditExecute

gotolimits.asm

```

1 #SIDIROPOULOS PANAGIOTIS AM:4489 cse84489
2
3 .data
4
5 str_nl: .asciz "\n"
6
7 .text
8
9 L0:
10     j main
11
12 goToLower:      # 1: begin_b
13     sw ra, -0(sp)
14
15 L2:              # 2: >, x, lower_lim
16     lw t0, -4(sp)
17     addi t0, t0, -12
18     lw t1, (t0)
19     lw t0, -4(sp)
20     addi t0, t0, -24
21     lw t2, (t0)
22     bgt t1, t2, L4
23
24 L3:              # 3: jump, _, _, 7
25     j L7

```

Line: 1 Column: 1 ☒ Show Line Numbers

MessagesRun I/O

```

5
0
10
-99999

-- program is finished running (0) --

5
12
10
-99999

-- program is finished running (0) --

5
2
10
-99999

-- program is finished running (0) --

```

Clear

EditExecute

gotolimits.asm

```

1 #SIDIROPOULOS PANAGIOTIS AM:4489 cse84489
2
3 .data
4
5 str_nl: .asciz "\n"
6
7 .text
8
9 L0:
10     j main
11
12 goToLower:      # 1: begin_b
13     sw ra, -0(sp)
14
15 L2:              # 2: >, x, lower_lim
16     lw t0, -4(sp)
17     addi t0, t0, -12
18     lw t1, (t0)
19     lw t0, -4(sp)
20     addi t0, t0, -24
21     lw t2, (t0)
22     bgt t1, t2, L4
23
24 L3:              # 3: jump, _, _, 7
25     j L7

```

Line: 24 Column: 25 ☒ Show Line Numbers

MessagesRun I/O

```

5
10
15
0

-- program is finished running (0) --

5
8
10
10

-- program is finished running (0) --

5
6
10
5

-- program is finished running (0) --

```

Clear

EditExecute

gotolimits.asm

```

1 #SIDIROPOULOS PANAGIOTIS AM:4489 cse84489
2
3 .data
4
5 str_nl: .asciz "\n"
6
7 .text
8
9 L0:
10     j main
11
12 goToLower:      # 1: begin_b
13     sw ra, -0(sp)
14
15 L2:              # 2: >, x, lower_lim
16     lw t0, -4(sp)
17     addi t0, t0, -12
18     lw t1, (t0)
19     lw t0, -4(sp)
20     addi t0, t0, -24
21     lw t2, (t0)
22     bgt t1, t2, L4
23
24 L3:              # 3: jump, _, _, 7
25     j L7

```

Line: 24 Column: 25 ☒ Show Line Numbers

MessagesRun I/O

```

10
11
3
-99999

-- program is finished running (0) --

```

Clear

Έλεγχος για την εμφάνιση των μεταβλητών και το πέρασμα παραμέτρων

Πρόγραμμα: nested.cpy

```
def main_F():
#{
    #declare glob

    def F1(param):
    #{
        #$ param = glob = 1 #$

        def F2():
        #{
            #declare overshadowed

            def F3():
            #{
                #declare var

                def F4():
                #{
                    #declare overshadowed
                    overshadowed = 0;
                    return(glob + param + var + overshadowed);
                #}
                var = 1;
                return(F4());
            #}

            overshadowed = glob;
            return(F3());
        #}

        return(F2());
    #}

    glob = 1;
    print(F1(glob));    #$ prints 3 #$
#}

if __name__ == "__main__":
    main_F();
```

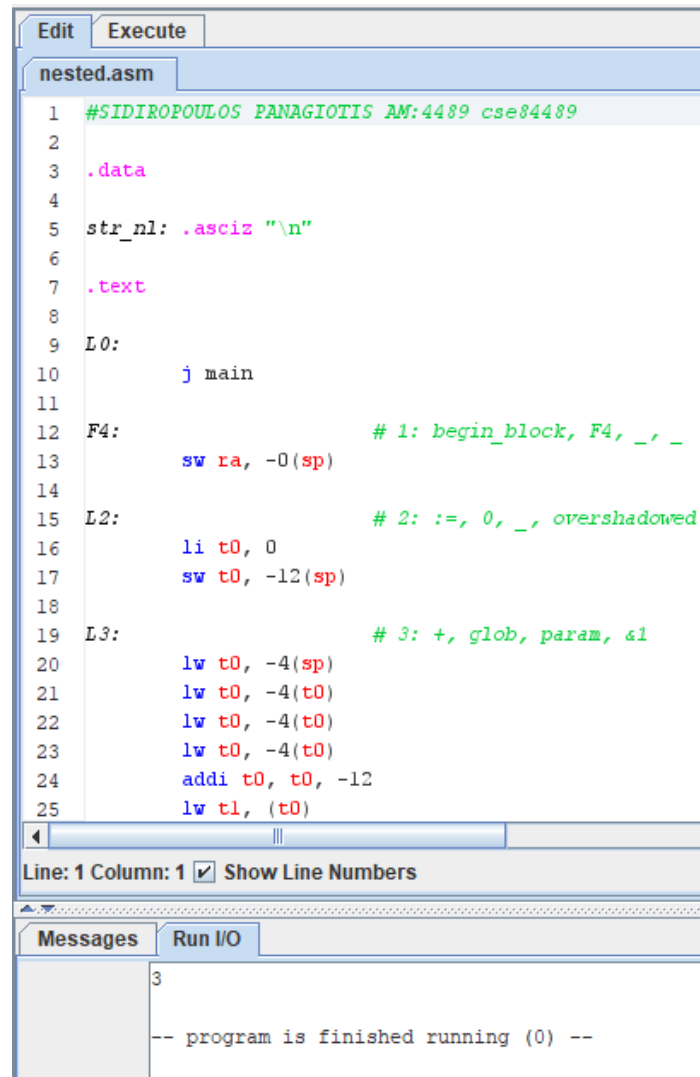
Λειτουργία:

1. Θέτει την καθολική μεταβλητή glob την τιμή 1
2. Περνάει την glob με τιμή στην παράμετρο param της F1
3. Θέτει την τοπική μεταβλητή της F2 overshadowed την τιμή της glob δηλαδή 1
4. Θέτει την τοπική μεταβλητή της F3 var την τιμή 1
5. Θέτει την τοπική μεταβλητή της F4 overshadowed (ίδιο όνομα με αυτής της F2) την τιμή 0
6. Υπολογίζει το άθροισμα των παραπάνω μεταβλητών και παραμέτρων και το τυπώνει στην κονσόλα

Το αποτέλεσμα που τυπώνεται στην κονσόλα είναι το 3 και αυτό μας επιβεβαιώνει ότι το πέρασμα των μεταβλητών και η εμφάνισή τους είναι σωστή καθώς γίνεται η επισκίαση της μεταβλητής overshadowed στην F4 όπως και υποστηρίζεται η πρόσβαση σε καθολικές και τοπικές μεταβλητές και παραμέτρους των συναρτήσεων που βρίσκονται σε χαμηλότερο επίπεδο φωλιάσματος από τις συναρτήσεις που βρίσκονται σε υψηλότερο .

Αναμενόμενο αποτέλεσμα: 3

Αποτέλεσμα εκτέλεσης:



The screenshot shows an assembly editor window titled 'nested.asm' with 'Edit' and 'Execute' tabs. The assembly code is as follows:

```
1 #SIDIROPOULOS PANAGIOTIS AM:4489 cse84489
2
3 .data
4
5 str_nl: .asciz "\n"
6
7 .text
8
9 L0:
10     j main
11
12 F4:                                # 1: begin_block, F4, _, _
13     sw ra, -0(sp)
14
15 L2:                                # 2: :=, 0, _, overshadowed
16     li t0, 0
17     sw t0, -12(sp)
18
19 L3:                                # 3: +, glob, param, &1
20     lw t0, -4(sp)
21     lw t0, -4(t0)
22     lw t0, -4(t0)
23     lw t0, -4(t0)
24     addi t0, t0, -12
25     lw t1, (t0)
```

Below the code editor, the status bar shows 'Line: 1 Column: 1' and a checked 'Show Line Numbers' option. At the bottom, there are 'Messages' and 'Run I/O' tabs. The 'Messages' tab is active, displaying the output:

```
3
-- program is finished running (0) --
```

Έλεγχος χρήσης αρνητικών μεταβλητών:

Πρόγραμμα: negativeVars.cpy

```
def main_negativeVars():
#{
    #declare neg_x,x

    def getThousand():
    #{
        return(1000);
    #}

    x = 5;
    neg_x = -x;

    if (neg_x < 0 and [neg_x == -x]):
        print(1);
    else:
        print(0);

    if (0 > -getThousand()):
        print(1);
    else:
        print(0);

    if (0 == (-getThousand() + 1000)):
        print(1);
    else:
        print(0);

#}

if __name__ == "__main__":
    main_negativeVars();
```

Λειτουργία:

Υλοποιεί μερικούς ελέγχους χρησιμοποιώντας αρνητικές μεταβλητές. Με την εκτέλεσή του αναμένουμε να τυπωθεί τρεις φορές ο αριθμός ένα

Αναμενόμενο αποτέλεσμα: Ο αριθμός 1 τυπωμένος στην κονσόλα 3 φορές

Αποτέλεσμα εκτέλεσης: (επόμενη σελίδα)

EditExecute

negativeVars.asm

```
1 #SIDIROPOULOS PANAGIOTIS AM:4489 cse84489
2
3 .data
4
5 str_nl: .asciz "\n"
6
7 .text
8
9 L0:
10     j main
11
12 getThousand:      # 1: begin_block, getThousand, _, _
13     sw ra, -0(sp)
14
15 L2:               # 2: ret, 1000, _, _
16     li t1, 1000
17     lw t0, -8(sp)
18     sw t1, (t0)
19     lw ra, (sp)
20     jr ra
21
22 L3:               # 3: end_block, getThousand, _, _
23     lw ra, (sp)
24     jr ra
25
26 main_negativeVars: # 4: begin_block, main_negativeVars, _, _
27     sw ra, -0(sp)
28
29 L5:               # 5: :=, 5, _, x
30     li t0, 5
31     sw t0, -16(sp)
```

Line: 1 Column: 1 ☒ Show Line Numbers

MessagesRun I/O

1
1
1

-- program is finished running (0) --

Έλεγχος μετατροπής κώδικα σε C:

Πρόγραμμα: demoToC.cpy

```
def main_firstToTen():
#{
    #declare a,b,i_a,i_b
    a = int(input());
    b = int(input());

    while (a < 10 and b < 10 ):
#{
    i_a = 0;
    i_b = 1;

    while (i_a < 5):
#{
        a = a + 1;
        i_a = i_a + 1;
    #}

    while (i_b > 0):
#{
        b = b * 2;
        i_b = i_b - 1;
    #}
#}

    if (a >= 10 and b >= 10):
#{
        print(22222);#$ draw #$
    #}
    else:
#{
        if (a >= 10):
#{
            print(0);    #$ a won #$
            print(a);
        #}
        else:
#{
            print(1);    #$ b won #$
            print(b);
        #}
    #}

#}
if __name__ == "__main__":
    main_firstToTen();
```

Λειτουργία:

Είσοδος από τον χρήστη το a και το b και υλοποιεί έναν αγώνα μέχρι το 10, με το "a" αυξάνεται κατά 5 και το "b" να διπλασιάζεται.

Αναμενόμενο αποτέλεσμα:

	Test1	Test2	Test3
a	1	-5	2
b	2	6	4
αποτέλεσμα	011	112	2222

Αποτέλεσμα εκτέλεσης: (επόμενη σελίδα)

```

1 //SIDIROPOULOS PANAGIOTIS AM:448
2
3 #include <stdio.h>
4
5 int main(){
6     int a, b, i_a, i_b, T_1, T_2
7     L_1: // begin_block, main
8     L_2: scanf("%d", &a); // in, a, _
9     L_3: scanf("%d", &b); // in, b, _
10    L_4: if (a < 10) goto L_6;
11    L_5: goto L_25; // jump, _
12    L_6: if (b < 10) goto L_8;
13    L_7: goto L_25; // jump, _
14    L_8: i_a = 0; // :=, 0, _
15    L_9: i_b = 1; // :=, 1, _
16    L_10: if (i_a < 5) goto L_12;
17    L_11: goto L_17; // jump, _
18    L_12: T_1 = a + 1; // +, a, 1
19    L_13: a = T_1; // :=, &1, _
20    L_14: T_2 = i_a + 1; // +, i_a, 1
21    L_15: i_a = T_2; // :=, &2, _
22    L_16: goto L_10; // jump, _
23    L_17: if (i_b > 0) goto L_19;
24    L_18: goto L_24; // jump, _
25    L_19: T_3 = b * 2; // *, b, 2
26    L_20: b = T_3; // :=, &3, _
27    L_21: T_4 = i_b - 1; // -, i_b, 1
28    L_22: i_b = T_4; // :=, &4, _
29    L_23: goto L_17; // jump, _
30    L_24: goto L_4; // jump, _
31    L_25: if (a >= 10) goto L_27;
32    L_26: goto L_31; // jump, _
33    L_27: if (b >= 10) goto L_29;
34    L_28: goto L_31; // jump, _
35    L_29: printf("%d", 22222);

```

1
2
011

...Program finished with exit code 0
Press ENTER to exit console.

```

18    L_12: T_1 = a + 1; // +, a, 1
19    L_13: a = T_1; // :=, &1, _
20    L_14: T_2 = i_a + 1; // +, i_a, 1
21    L_15: i_a = T_2; // :=, &2, _
22    L_16: goto L_10; // jump, _
23    L_17: if (i_b > 0) goto L_19;
24    L_18: goto L_24; // jump, _
25    L_19: T_3 = b * 2; // *, b, 2
26    L_20: b = T_3; // :=, &3, _
27    L_21: T_4 = i_b - 1; // -, i_b, 1
28    L_22: i_b = T_4; // :=, &4, _
29    L_23: goto L_17; // jump, _
30    L_24: goto L_4; // jump, _
31    L_25: if (a >= 10) goto L_27;
32    L_26: goto L_31; // jump, _
33    L_27: if (b >= 10) goto L_29;
34    L_28: goto L_31; // jump, _
35    L_29: printf("%d", 22222);

```

-5
6
112

...Program finished with exit code 0
Press ENTER to exit console.

```

Edit Execute
demoToC.asm
1 #SIDIROPOULOS PANAGIOTIS AM:4489 cse84489
2
3 .data
4
5 str_nl: .asciz "\n"
6
7 .text
8
9 L0:
10     j main
11
12 main_firstToTen: # 1: be
13     sw ra, -0(sp)
14
15 L2: # 2: in, a, _
16     li a7, 5
17     ecall
18     sw a0, -12(sp)
19
20 L3: # 3: in, b, _
21     li a7, 5
22     ecall
23     sw a0, -16(sp)
24
25 L4: # 4: <, a, 10, 6

```

Line: 1 Column: 1 Show Line Numbers

Messages Run I/O

1
2
0
11

-- program is finished running (0) --

-5
6
1
12

-- program is finished running (0) --

2
4
22222

-- program is finished running (0) --

Clear