**141116 zh3 minta**

**1. feladat:**

Mellékelten adott egy blokkstrukturált nyelven írt forrásprogram azon fordítási pillanatát tükröző szimbólumtáblája, amikor a ***citrom*** *eljárás* törzsének fordítására kerül sor.

1. Elemezze szemantikusan, hogy a ***citrom***eljárás törzsében helyes-e a következő utasítás! Tegyük fel, hogy szigorúan típusos a nyelv.

**d[2]:= c + b[max];**

1. Elemezze, hogy az ***alma*** program törzsében, ahol már nem érvényesek a *barack* és a *citrom* eljárás lokális deklarációi, helyes-e az alábbi utasítás!

**e:=barack(d);**

1. feladat táblája:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Szimbólumtábla | |  |  |  |  |  |  |  |
| cím | név | fajta | életjel | érték | típus leíró | ráköv. | parák száma | első paraméter |
| 1 | integer | typedef | true |  | 100 |  |  |  |
| 2 | real | typedef | true |  | 200 |  |  |  |
| 3 | alma | progdef | true |  | nil |  | 0 |  |
| 4 | max | constdef | true | 10 | 100 |  |  |  |
| 5 | korte | typedef | true |  | 300 |  |  |  |
| 6 | a | vardef | true |  | 100 |  |  |  |
| 7 | c | vardef | true |  | 100 |  |  |  |
| 8 | d | vardef | true |  | 300 |  |  |  |
| 9 | e | vardef | true |  | 200 |  |  |  |
| 10 | barack | funcdef | true |  | 100 |  | 1 | 11 |
| 11 | c | varparadef | true |  | 300 | nil |  |  |
| 12 | a | vardef | true |  | 100 |  |  |  |
| 13 | b | vardef | true |  | 200 |  |  |  |
| 14 | citrom | procdef | true |  | nil |  | 0 |  |
| 15 | c | vardef | true |  | 200 |  |  |  |
| 16 | b | vardef | true |  | 300 |  |  |  |
| Blokkindex vektor | |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |
| 11 |  |  |  |  |  |  |  |  |
| 15 |  |  |  |  |  |  |  |  |
| Típusleíró tábla | |  |  |  |  |  |  |  |
| cím | fajta | név | méret | index-leíró | elemleíró | min | max |  |
| 100 | skalar | 1 |  |  |  | -32768 | +32767 |  |
| 200 | skalar | 2 |  |  |  |  |  |  |
| 300 | tömb | 5 | 6 | 400 | 200 |  |  |  |
| 400 | inter-vallum |  |  |  | 100 | 5 | 10 |  |

**1. feladat megoldása:**

**d[2]:= c + b[max];** értékadás ellenőrzése**:**

8-as címen szerepel d, típusa 300, ami egy tömb 5-től 10-ig indexelve. A 2-es index hibás.

A tömb elemeinek típusa 200 (real), így az értékadás baloldala 200-as típusú.

15-ön szerepel c, típusa 200.

16-on szerepel b, típusa 300.

4-en szerepel max, típusa 100, érteke 10, így megfelel b indexének.

b[max] típusa 200, így az összeadás mindkét operandusa 200 (real), így ez helyes, és a kifejezés típusa is 200.

Mivel az értékadása baloldala 200, a jobboldala 200, így az értékadás is helyes.

**e:=barack(d);** ellenőrzése:

11-től 16-ig az életjelek hamisak.

9-en szerepel e, típusa 200.

10-en szerepel barack, típusa 100, így a szigorú típusosságnak nem felel meg az értékadás.

Egyébként a barack 1 paramétert vár 300-as típussal.

8-on szerepel d, típusa 300, így a függvényhívás helyes.

**SZIMBOLUM\_MINTA.DOC**

**1.feladat:**

Mellékelten adott egy blokkstruktúrált nyelv azon fordítási pillanatát tükröző szimbólumtáblája, amikor a *tigris* függvény törzsének fordítására kerül sor.

a) Elemezd szemantikusan, hogy a *tigris* függvény törzsében helyes-e a következő utasítás:

**munka[nap]:= n** .

b) Elemezd, hogy a *macko* program törzsében helyesek-e az alábbi utasítások:

**dolog[szombat]:=15;**

**tigris(munka,nap) .**

c) Tegyük fel, hogy a tigris függvény után deklarálva van egy **malacka** paraméter nélküli eljárás, amelynek van egy **i : byte** és egy **napok: munka** lokális változója. Rajzold át a szimbólumtáblát úgy, hogy ellenőrizni tudd a malacka eljárás törzsében szereplő

**tigris(napok , i)**

utasítást.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Szimbólum tábla | |  |  |  |  |  |  |  |
| cím | név | típus | életjel | érték | típus leíró | ráköv. | paraméterek száma | első paraméter |
| 1 | byte | typedef |  |  | 100 |  |  |  |
| 2 | integer | typedef |  |  | 200 |  |  |  |
| 3 | macko | progdef |  |  | nil |  | 0 |  |
| 4 | napok | typedef |  |  | 300 |  |  |  |
| 5 | hetfo | constdef |  | 0 |  | 6 |  |  |
| 6 | kedd | constdef |  | 1 |  | 7 |  |  |
| 7 | szerda | constdef |  | 2 |  | 8 |  |  |
| 8 | csutortok | constdef |  | 3 |  | 9 |  |  |
| 9 | pentek | constdef |  | 4 |  | nil |  |  |
| 10 | nap | vardef |  |  | 300 |  |  |  |
| 11 | munka | typedef |  |  | 400 |  |  |  |
| 12 | dolog | vardef |  |  | 400 |  |  |  |
| 13 | tigris | funcdef |  |  | 200 |  | 2 | 14 |
| 14 | munka | varpara |  |  | 400 | 15 |  |  |
| 15 | n | constpara |  |  | 100 | nil |  |  |
| 16 | i | vardef |  |  | 300 |  |  |  |
| Blokkindex vektor | |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |
| 14 |  |  |  |  |  |  |  |  |
| Típusleíró tábla | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
| cím | típus | név | indexleíró | elemleíró | min | max | első eleme |  |
| 100 | skalár | 1 |  |  | 0 | 255 |  |  |
| 200 | skalár | 2 |  |  | -32000 | 32000 |  |  |
| 300 | felsorolás | 4 |  |  | 0 | 4 | 5 |  |
| 400 | tömb | 12 | 300 | 100 |  |  |  |  |

(no meo)

**141116 zh3 minta**

**2. feladat:**

Adott az alábbi C++ kódrészlet, ahol egy blokk kapcsos zárójeltől kapcsos zárójelig tart, illetve a for ciklus is külön blokkot képez.

**int main()**

**{**

**int i=3;**

**string t="nem";**

**const int n=10;**

**{**

**int a[n];**

**for (int i=8; i<=10; i++) a[i]=t;**

**cout << i << endl;**

**if (0==(i % 2))**

**t="igen";**

**else**

**{ char t=nem;}**

**cout<<t<<endl;**

**}**

**return 0;**

**}**

1. Építsen hozzá szimbólumtáblát, amely a záró kapcsos zárójelig mutatja a tábla alakulását!

A már nem élő szimbólumokat fizikailag nem vesszük ki a táblából, csak hamisra állítjuk az életjelüket.

1. Milyen szemantikus hibákat jelez a fordítóprogram?
2. Ha ignoráljuk a hibás sorokat, akkor mit ír ki a program?

**2. feladat megoldása:**

1. A szimbólumtábla a következő oldalon látható.
2. Az a[i]=t értékadás hibás, mert a baloldal int típusú, míg a jobboldal string típusú.  
   A t=nem értékadás hibás, mert a *nem* szimbólum nincs definiálva.
3. Ha a hibás értékadásoktól eltekintünk, akkor a program a következőt írja ki:

3 // main()-ben deklarált i értéke

nem // i páratlan, így szintén a main()-ben deklarált t értéke nem változik

2. feladat táblája:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Szimbólumtábla | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
| cím | név | fajta | életjel | érték | típus leíró | ráköv. | parák száma | első paraméter |
| 1 | int | typedef | true |  | 100 |  |  |  |
| 2 | char | typedef | true |  | 200 |  |  |  |
| 3 | string | typedef | true |  | 300 |  |  |  |
| 4 | main | funcdef | true |  | 100 |  |  |  |
| 5 | i | vardef | false |  | 100 |  |  |  |
| 6 | t | vardef | false |  | 300 |  |  |  |
| 7 | n | vardef | false | 10 | 100 |  |  |  |
| 8 | a | vardef | false |  | 500 |  |  |  |
| 9 | i | vardef | false |  | 100 |  |  |  |
| 10 | t | vardef | false |  | 200 |  |  |  |
| 11 |  |  |  |  |  |  |  |  |
| Blokkindex vektor | |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |
| ~~5~~ |  |  |  |  |  |  |  |  |
| ~~8~~ |  |  |  |  |  |  |  |  |
| ~~9~~ | ~~10~~ |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
| Típusleíró tábla | |  |  |  |  |  |  |  |
| cím | fajta | név | méret | index-leíró | elemleíró | min | max |  |
| 100 | skalar | 1 |  |  |  | -32768 | +32767 |  |
| 200 | skalar | 2 |  |  |  | 0 | 255 |  |
| 300 | tömb | 5 | 256 | 400 | 200 |  |  |  |
| 400 | inter-vallum | - |  |  | 100 | 0 | 255 |  |
| 500 | tömb | - | 10 | 600 | 100 |  |  |  |
| 600 | inter-vallum | - |  |  | 100 | 0 | 9 |  |

**140103 zh4 pót**

**1. feladat:**  Hogyan  állítja az **ADD BX,1** utasítás az **O, S, Z, C flegeket,** ha BX=7FFFh? (4 pont)

Megoldás: 7FFF O=1, S=1, Z=0, C=0

+0001

8000

// pl ha -128 tol vannak a számok 127 ig, És te 127 hez 1-et adsz, Akkor -128at kapsz,

// És ilyenkor a carry 1

// Mert előjelet váltott a szám, Pedig normál esetben nem kellett volna neki, Mert 127+1 az 128

// de ez egyben oweflow is \_itt\_

// de akkor a kettő fleg között mi a különbség? Az egyik az előjeles túlcsordulás, A másik az előjel nélküli

**zh4 minta**

2./ Hogyan állítja az ADD BL,CH utasítás a **Z** (zero) és az **C** (carry) flegeket, ha a regiszterek érteke ECX=A0B0C0D0h, BX=1234h ? (4 pont)

Megoldás: BL=34h és CH=C0h összege F4h.   
 Tehát nem keletkezett carry, azaz C=0 és nem zéró az eredmény, azaz Z=0.

**140103 zh4 pót**

**2. feladat:** Mi a hasonlóság és a különbség az alábbi programrészletek végrehajtása között? (4 pont)

XOR EAX,EAX MOV EAX,0

NOT EAX SUB EAX,1

Megoldás: Mindkét részlet végrehajtása után EAX-ben minden bit 1-es lesz.

Az első részlet ezt logikai műveletekkel éri el, a másik esetben nullából von ki 1-est, aminek -1 az eredménye, ami kettes komplemens kódban szintén csupa 1-es bitet jelent.

**zh4 minta**

1./ Milyen logikai művelettel lehetne úgy tesztelni az AL regisztert, hogy AL tartalma ne változzon, de megtudjuk, hogy kettes komplemenskódban negatív szám van-e benne?

Melyik fleg értéke mutatná a teszt eredményt? (4 pont)

Megoldás: Akár az OR AL,AL , akár az AND AL,AL utasítással tesztelhető. A művelet eredményeként a szignum fleg (S fleg) mutatja AL előjelét.

**140103 zh4 pót**

**3. feladat:** Mi lesz az AX regiszter tartalma az alábbi programrészlet végrehajtása után? (6 pont)

section .data

alma DW -1, 1

section .text

global main

main

MOV AX, [alma]

CMP AX,[alma+2]

JG nagyobb

JMP vege

nagyobb: XOR AX,AX

vege: ….

Megoldás: AX = -1 , mert -1 előjelesen nem nagyobb, mint 1, így nem hajtódik végre a XOR.

|  |  |  |
| --- | --- | --- |
| **Directive** | **Purpose** | **Storage Space** |
| DB | Define Byte | allocates 1 byte |
| DW | Define Word | allocates 2 bytes |
| DD | Define Doubleword | allocates 4 bytes |
| DQ | Define Quadword | allocates 8 bytes |
| DT | Define Ten Bytes | allocates 10 bytes |

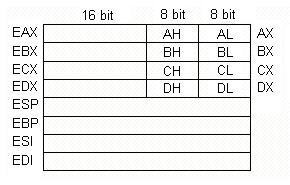
**zh4 minta**

3./ Mi lesz a **BH** regiszter tartalma a **MOV BX,[adat]** utasítás végrehajtása után?

Ahol **adat DB 1,2,3,4,5,6,7,8**

(2 pont)

Megoldás: BX=0201h, mert byte-fordított sorrend van, így BH=02h.

****

**140103 zh4 pót**

**4. feladat:** Mely utasítások helyesek, és melyek nem helyesek az alábbiak közül és miért? (5 pont)

Tegyük fel, hogy az utasításokban szereplő címkék léteznek.

* + 1. MOV [KORTE],[ALMA]
    2. PUSH EAX,EBX
    3. MOV EBX, 0AH
    4. MOV byte [ALMA],320
    5. POP word[ALMA]

Megoldás:

1. Hibás, mert két memóriahivatkozás van benne.
2. Hibás, mert a PUSH műveletnek csak egy operandusa van.
3. Helyes, mert EBX-be 10-et tölt.
4. Hibás, mert egy byte-on nem ábrázolható 320.
5. Helyes.

**zh4 minta**

4./ Miért nem helyesek az alábbi utasítások? (4 pont)

**MOV word [adat],AH**

**SUB [EAH],10B**

Megoldás: MOV operandusai eltérő méretűek (16 bit, 8 bit).

SUB esetén [EAH] egy memóriahivatkozás,   
10B egy bináris konstans. Így nincs megadva az operandusok mérete.

**140103 zh4 pót**

**5. feladat:** a) Mi kerül az „eredemeny” változóba a program futása során? (3 pont)

b) Melyik műveletet valósítja meg az „eljárás”? (2 pont)

global main

extern ki\_egesz

section .bss

eredmeny resd 1

section .data

a dd 6

b dd 10

section .text

main: push dword [b]

push dword [a]

call eljaras

add esp,8

mov [eredmeny],eax

push dword [eredmeny]

call ki\_egesz

add esp,4

ret

eljaras:

mov ebx,[esp+4] ; 2.paraméter

cmp ebx,0

je ugrik

xor edx,edx

mov eax,[esp+8] ; 1.paraméter

div ebx

mov eax,edx

ret

ugrik: mov eax,-1

ret

Megoldás: Az eredmeny=4, amit ki is ír a főprogram. Az eljárás a „b mod a” –t hajtja végre, azaz az osztás maradékát adja vissza, ha nem nulla az osztó. Ha nulla az osztó, akkor -1 –t ad vissza. A konkrét hívás esetén 10 mod 6 –ot adja vissza.

**zh4 minta**

5./ a) Mi kerül az „eredemeny” változóba a program futása során? (2 pont)

b) Melyik függvényt valósítja meg az „eljárás” ? Add meg a rekurzív függvény definícióját! (4 pont)

section .bss

eredmeny resd 1

section .txt

global main

main: push 4

call eljaras

add esp,4

mov [eredmeny],eax

ret

; Az eljárás egy függvényt valósít meg, egy paramétere van, a visszaadott érték eax-be kerül.

eljaras: cmp dword [esp+4],0 ; paraméter hasonlítása

ja tovabb

mov eax,0

ret

tovabb: mov ebx, [esp+4]

dec ebx

push ebx

call eljaras

pop ebx

inc ebx

add eax,ebx

ret

Megoldás: unsigned eljaras(unsigned n)

{

if (n>0)

{

return eljaras(n-1)+n

}

else return 0;

}

Ez lenne a megfelelő C kód.

Így eredmeny=eljaras(4)=0+1+2+3+4=10.

A szumma függvényt valósítja meg az eljárás, azaz

eljaras(n) = 0, ha n=0

eljárás(n) = eljaras(n-1)+n , ha n>0.

**FPÉLDA\_ATTR.DOC**

Adott az alábbi attribútum grammatika, amely egy értékadó utasítás szintaxisát és szemantikáját adja meg. Ennek segítségével ellenőrizd, hogy az alábbi értékadó utasítások helyesek-e! Az utolsó utasítás szintaxis fáját is add meg a megfelelő attribútum értékekkel!

a$ := a + b

a := alma$ + b$

a := a% + b

a := a + b%

a$ := a$ + b$ + c$

1./ <értékadás> ::= <azonosító> **:=** <kifejezés>

Condition: Type(<azonosító>) = Type(<kifejezés>)

2./ <kifejezés> ::= <kifejezés2> + <azonosító>

Type(<kifejezés>) <--- Type(<azonosító>)

Condition: (Type(<kifejezés2>) = Type(<azonosító>)) vagy

(Type(<kifejezés2>) ='integer' és Type(<azonosító>)='real') vagy

(Type(<kifejezés2>) ='real' és Type(<azonosító>)='integer')

| <azonosító>

Type(<kifejezés>) <--- Type(<azonosító>)

3./ <azonosító> ::= <betű>

Type(<azonosító>) <--- 'real'

| <betű>**%**

Type(<azonosító>) <--- 'integer'

| <betű>**$**

Type(<azonosító>) <--- 'string'

4./ <betű> ::= **a | ... | z**

**SZEMANTIKUS\_FELADAT.DOC**

**Feladat:**

1. Ellenőrizze a megadott attribútum grammatika segítségével az alábbi programot szemantikusan! Mondja meg, milyen hibaüzenetek várhatók!
2. Rajzolja fel az utolsó utasítás szintaxisfáját, adja meg az egyes csomópontokhoz tartozó attribútum értékeket és a megfelelő pontoknál, értékelje ki a feltételeket!

**BEGIN**

**CHAR x**

**STRING y**

**z = ”alma”**

**EQ x,’a’ : CONS y, x**

**END**