Vinicius Francisco da Silva - 576920

Lista III

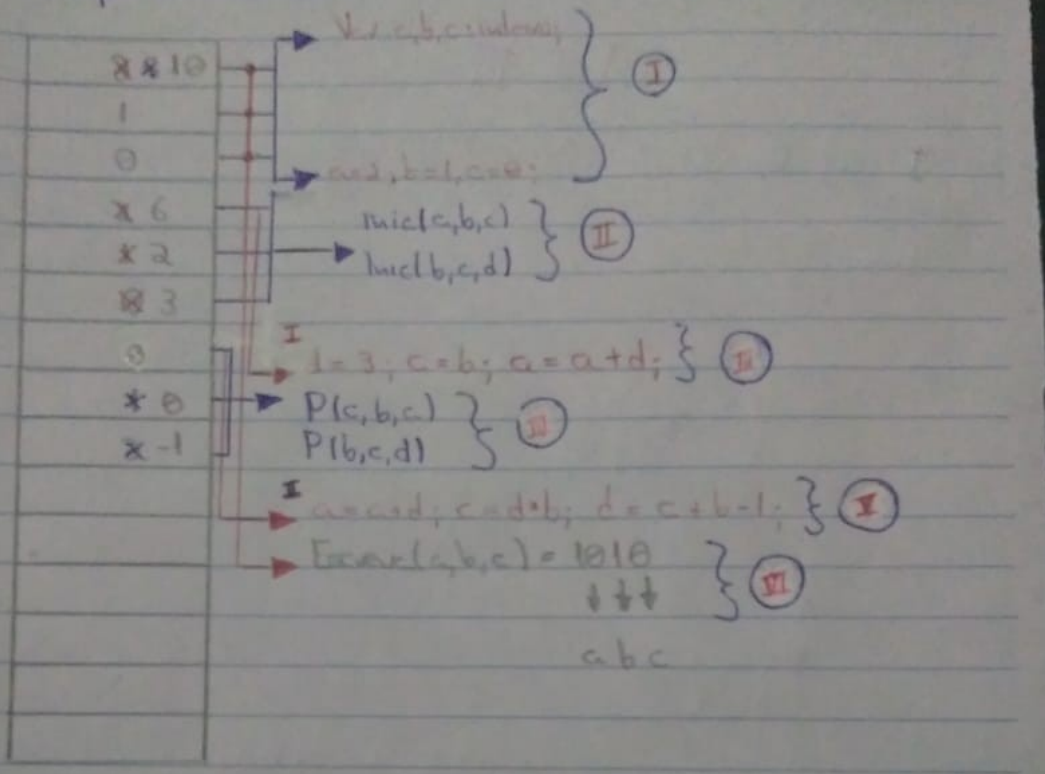
I a) Por valor

Execução

a = 2; b = 1; c = 0;
Inic(2, 1, 0)
Inic(b, c, d)
d = 3; c = b; b = 6; a = a + d;
Fim;
P(0, 1, 5)
P(c, b, a)
a = a + d; c = d * b; d = c + b - 1;
Fim;
Escreve(1, 0)
Saída = 1 0

a b c

Mapa de memória



(I) Reserva do espaço de memória para as variáveis inteiras a,b,c

(II) Passagem dos valores de a,b,c para espaços alocados referente aos parâmetros b,c,d

(III) Atualização dos valores das variáveis locais e atualização do espaço de memória equivalente a variável global a.

(IV) Passagem dos valores de c,b,a para espaços alocados referentes aos parâmetros b,c,d

tilibra (I) Atualização das variáveis locais e atualização da variável global a.

(II) Escrita dos valores

b) Por referencia

a=2 ; b=1 ; c=0

Inic(a,b,c) ;

Inic(b,c,d) ;

{ a=c+d ; c=d*b ; d=c+b-1

}

Fim ;

P(c,b,a) ;

P(b,c,d) ;

{ a=c+d ; c=d*b ; d=c+b-1 ;

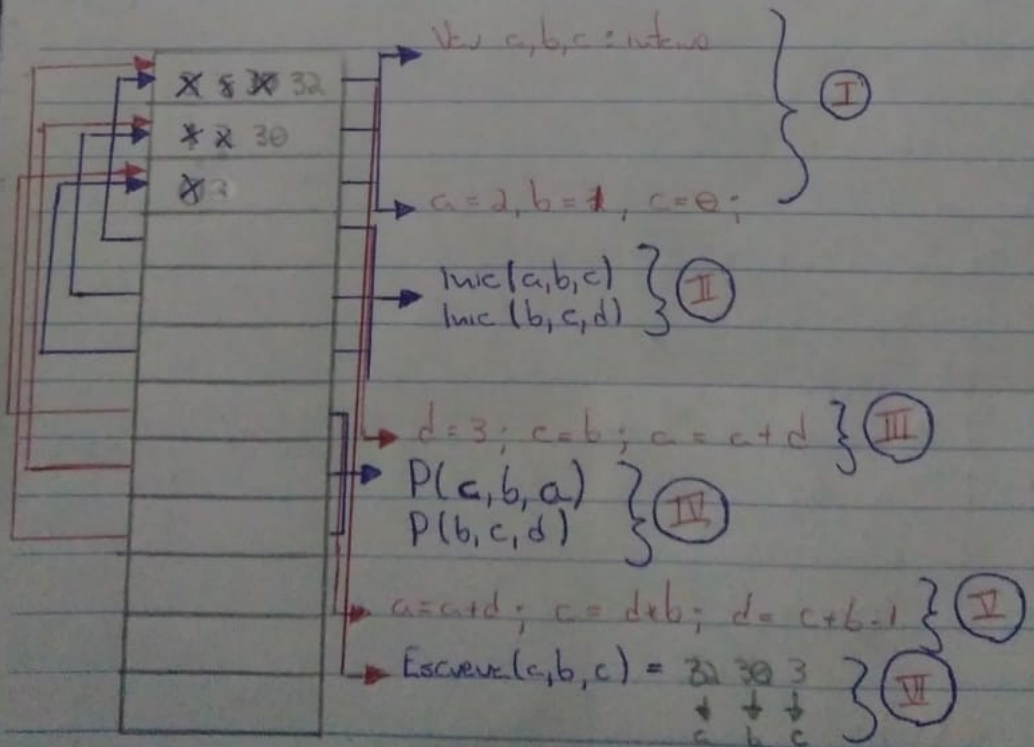
}

Fim

Escreve(c,b,c)

Saída = 32 30 3

a b c



(I) Reserva do espaço de memória para as variáveis inteiras a, b, c

(II) Passagem por referência das variáveis a, b, c.

(III) Atualização das variáveis passadas ao método por referência.

(IV) Passagem por referência das variáveis a, b, c.

(I) Atualização das variáveis passadas ao método por referência

(VI) Escrita dos valores

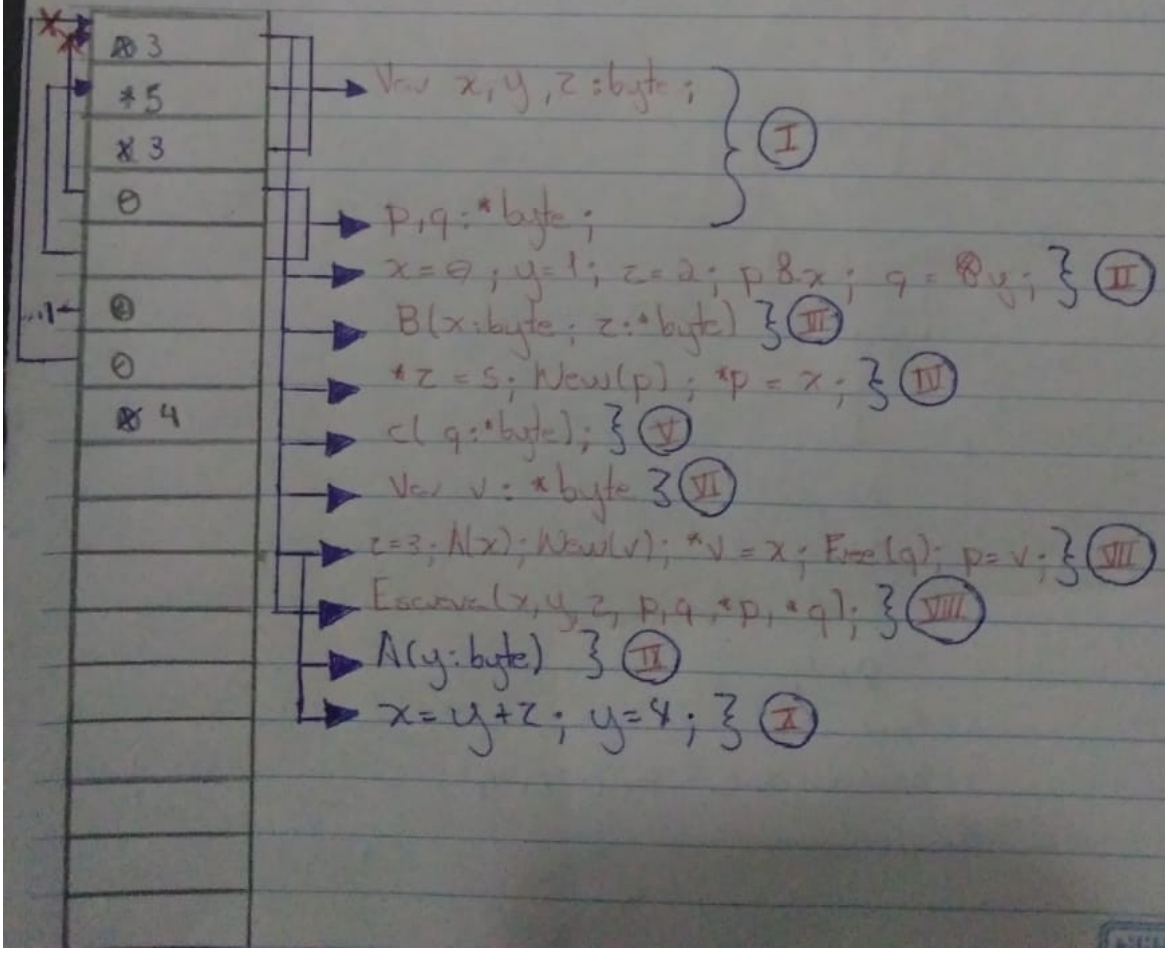
II a) Passagem por valor

Execução

```
Var x, y, z: byte;
p, q: ^byte;
Início
| x=0; y=1; z=2;
| p=@x; q=@y;
| B(y, q);
| | B(x:byte; z:^byte)
| | ^z=5;
| | New(p);
| | ^p=x;
| Fim;
| C(p);
| C(q:^byte);
| |
```

tilibra

```
Var v:*byte;
  c:byte;
Inicio
  z=3
  A(x)
  New(r);
  *v=x;
  Free(q);
  p=v;
Fin;
Escreve(x,y,z,p,q,*p,*q);
Fim;
```



I) Alocação dos espaços referentes as variáveis x, y, z do tipo byte, e das variáveis p, q referentes ao tipo apontador de byte.

II) Atribuição de valores nos espaços x, y, z e atribuição dos endereços de x e y em p e q, que são apontadores.

III) Passagem dos valores para os argumentos do método B.

IV) Atribuição do valor no endereço ~~em~~ que z esta apontado alocação do apontador p, atribuição de x onde p está apontando.

V) Passagem do valor de p para o método C.

VI) Criação de um apontador do tipo byte v.

VII) Atribuição de um valor em z, Passagem do valor de x no parametro A, atribuição do valor de x onde v esta apontando, liberação do espaço de q, atribuição do valor de v em p.

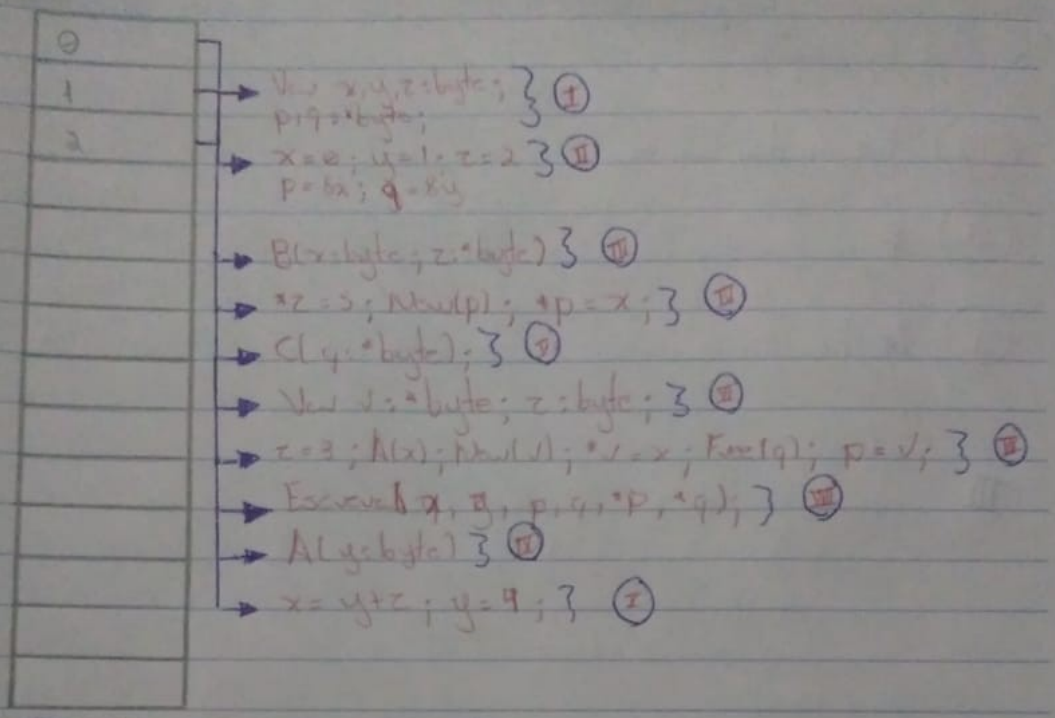
VIII) Escrita dos valores.

IX) Passagem do valor de x no argumento de A.

X) Alteração nos valores de x e da variavel local de y.

b) Passagem por referência

```
Execução
  Var x, y, z : byte;
  p, q : ^byte;
  Início
  | x := 0; y := 1; z := 2;
  | p := &x; q := &y;
  | B(y, q);
  | B(x : byte; z : ^byte)
  |   | ^z := 5;
  |   | New(p);
  |   | ^p := x;
  | Fim;
  | C(p);
  | C(q : ^byte);
  |   | Var v : ^byte;
  |   |   z : byte;
  | Início
  |   | z := 3;
  |   | A(x);
  |   | New(v);
  |   | ^v := x;
  |   | Free(q);
  |   | p := v;
  | Fim;
  | Escreva(x, y, z, p, q, ^p, ^q);
  Fim
```



(I) Reserva de espaços de memória para x, y, z e p, q como um ponteiro de byte.

(II) Atribuição de valores em x, y, z e o endereço de x e y em p e q

(III) Passagem dos valores → por referência para os argumentos do método B por referência.

(IV) Atribuição do valor → por referência constante para o endereço que z está apontando, p é alocado como um ponteiro, endereço onde v aponta recebe x, libera o espaço q.

(V) Passagem dos valores para os argumentos do método B por referência

(VI) Criação dos espaços equivalentes a v e z.

(VII) Atribuição de uma constante em z, Passagem de x para o método A, alocação do espaço v, atribuição do valor de x

tilibra

onde v aponta, liberação do espaço de memória onde q está, atribuição do valor de v em p.

(VIII) Escrita dos valores

(II) Passagem dos valores em A.

(X) Atualização das variáveis de x e y.

III
```
dseg SEGMENT PUBLIC
     byte 4000h DUP(9)
     sword 0
     byte cx, DUP(bx)
dseg ENDS
cseg SEGMENT PUBLIC
ASSUME CS:cseg, DS:dseg
stt
mov cx, dseg
mov ds, cx
mov ax, DS:[ta.end]
mov bx, 4
mul bx
mov DS:[ta.end], cx
mov cx, DS:[ta.end]
mov bx, DS:[tb.end]
add cx, bx
mov DS:[tb.end], cx
```
RI: 19 12

b)

```
dseg SEGMENT PUBLIC
  byte 4000h DUP(?)
  sword 8
dseg ENDS
cseg SEGMENT PUBLIC
ASSUME CS: cseg, DS: dseg
strt
mov ax, dseg
mov ds, ax
mov ax, DS:[i.end]
add ax, 10
mov DS:[t2.end], ax
mov cx, DS:[t1.end]
mov bx, DS:[t3.end]
add cx, bx
mov DS:[t5.end], cx
mov DS:[t4.end], bx
mov DS:[t9.end], cx
mov cx, DS:[x.end]
mov bx, DS:[t4.end]
mul cx, bx
mov DS:[t5.end], cx
mov cx, DS:[1.end]
add cx, 20
mov DS:[t6.end], cx
mov bx, DS:[t6.end]
mul bx, 8
mov DS:[t7.end], bx
mov DS:[t8.end], t7.end
```

b)

```
dseg SEGMENT PUBLIC
    byte 4000h DUP(?)
    sword 8
dseg ENDS
cseg SEGMENT PUBLIC
ASSUME CS: cseg, DS: dseg
strt
mov ax, dseg
mov ds, ax
mov ax, DS:[i.add]
add ax, 10
mov DS:[t2.end], ax
mov ax, DS:[t1.end]
mov bx, DS:[t2.end]
add ax, bx
mov DS:[t3.end], ax
mov DS:[t2.end], bx
mov DS:[t4.end], ax
mov ax, DS:[x.end]
mov bx, DS:[t4.end]
mul ax, bx
mov DS:[t5.end], ax
mov ax, DS:[1.end]
add ax, 20
mov DS:[t6.end], ax
mov bx, DS:[t5.end]
mul bx, 8
mov DS:[t7.end], bx
mov DS:[t8.end], t7.end
```

tilibra

```
mov ax, DS:[ts.ent]
mov bx, DS:[t8.end]
add ax, bx
mov DS:[t9.end], ax
~~cout t8,t2~~
mov ax, t7.end
mov bx, t9.end
mov ax, bx
mov DS:[t7.end], ax
mov ah, 4ch
int 21h
cseg ENDS
END start
```