RAM

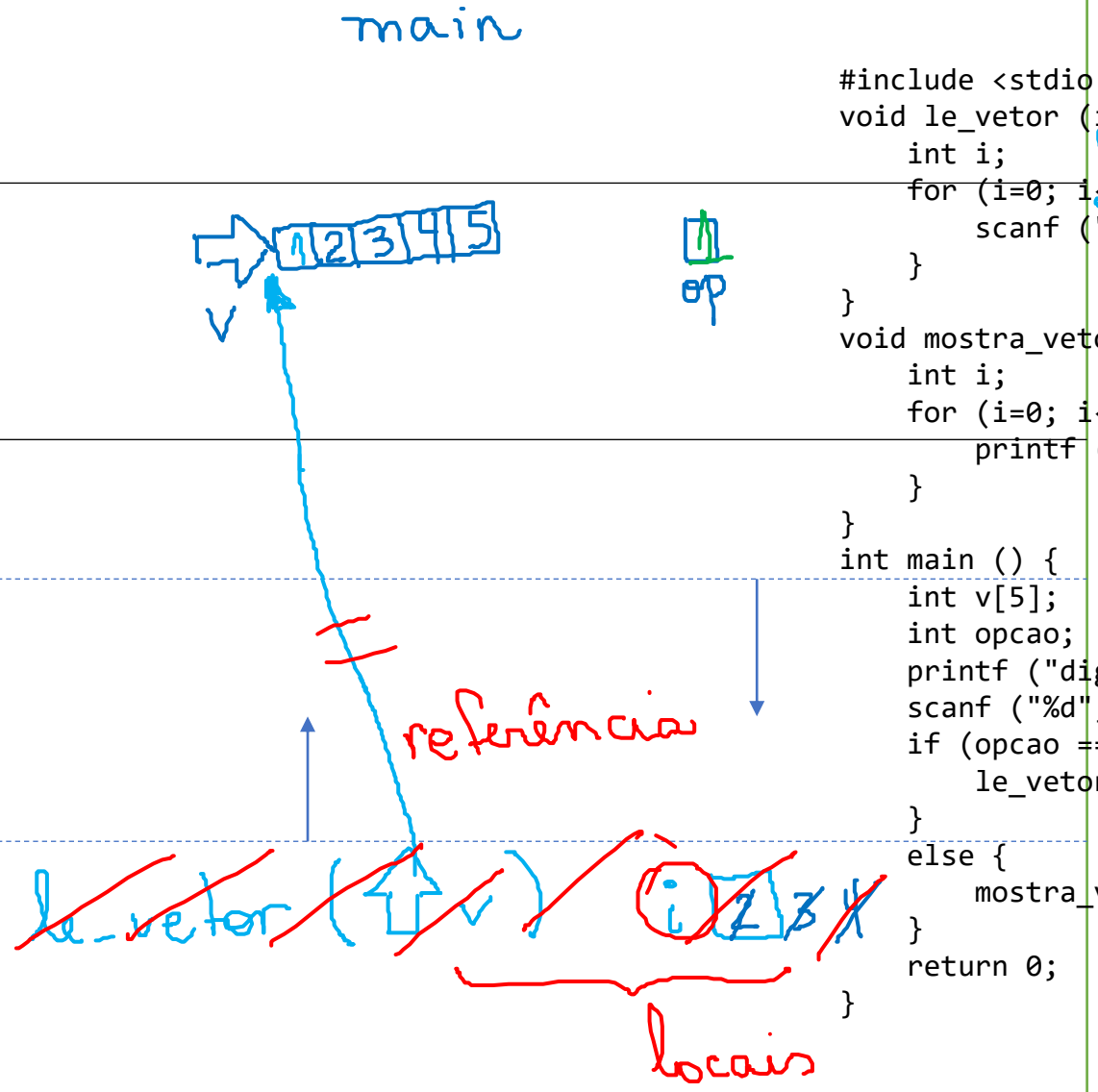
SO + aplicativos (principal)

Variáveis alocadas estaticamente + globais

Var. alocadas dinamicamente

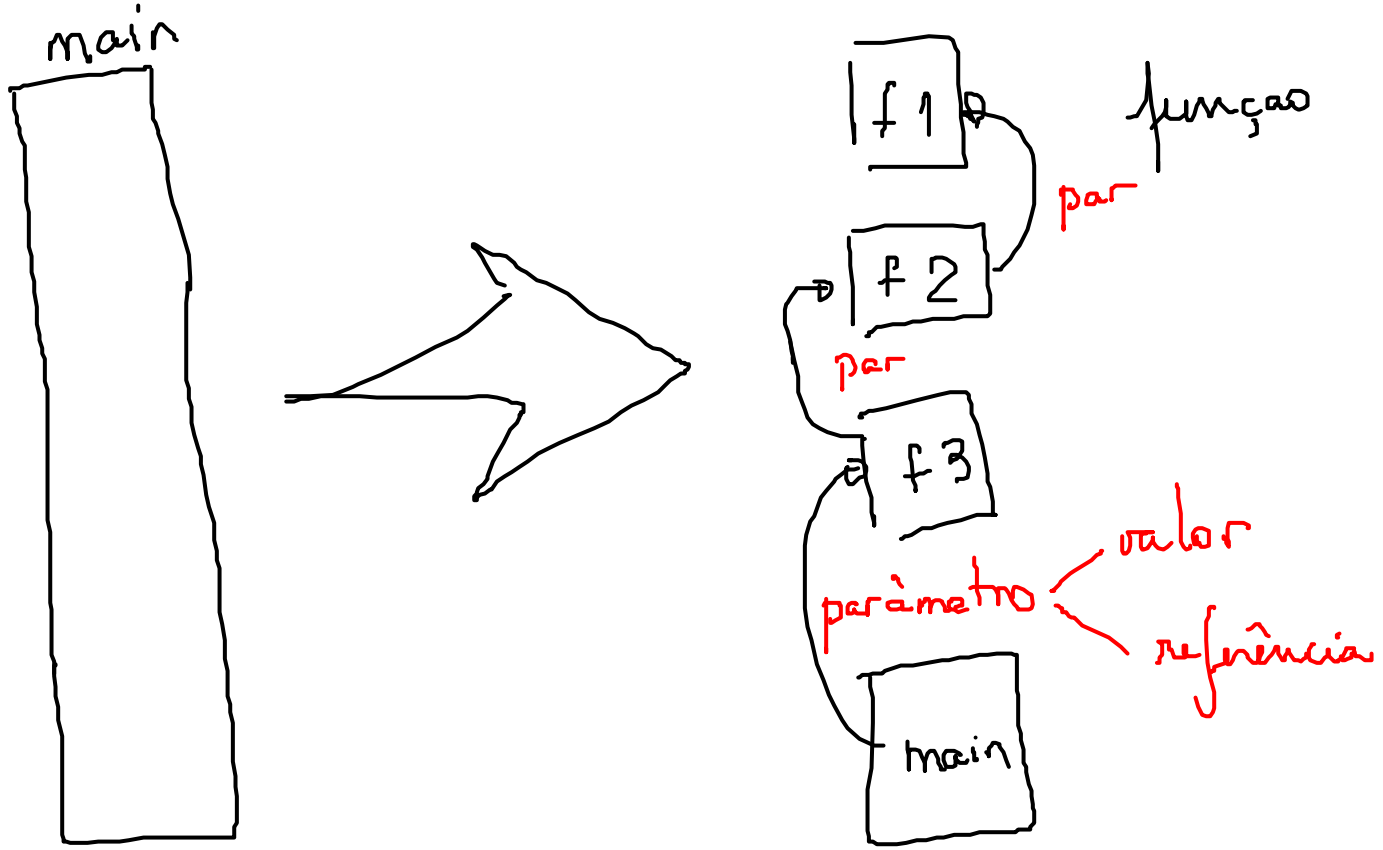
livre

Pilha de execução




```c
#include <stdio.h>
void le_vetor (int v[]) {
    int i;
    for (i=0; i<5; i++) {
        scanf ("%d", &v[i]);
    }
}
void mostra_vetor (int v[]) {
    int i;
    for (i=0; i<5; i++) {
        printf ("%d\n", v[i]);
    }
}
int main () {
    int v[5];
    int opcao;
    printf ("digite 1 para ler vetor ou 2 para exibir vetor");
    scanf ("%d", &opcao);
    if (opcao == 1) {
        le_vetor(v);
    }
    else {
        mostra_vetor(v);
    }
    return 0;
}
```

modularização



RAM

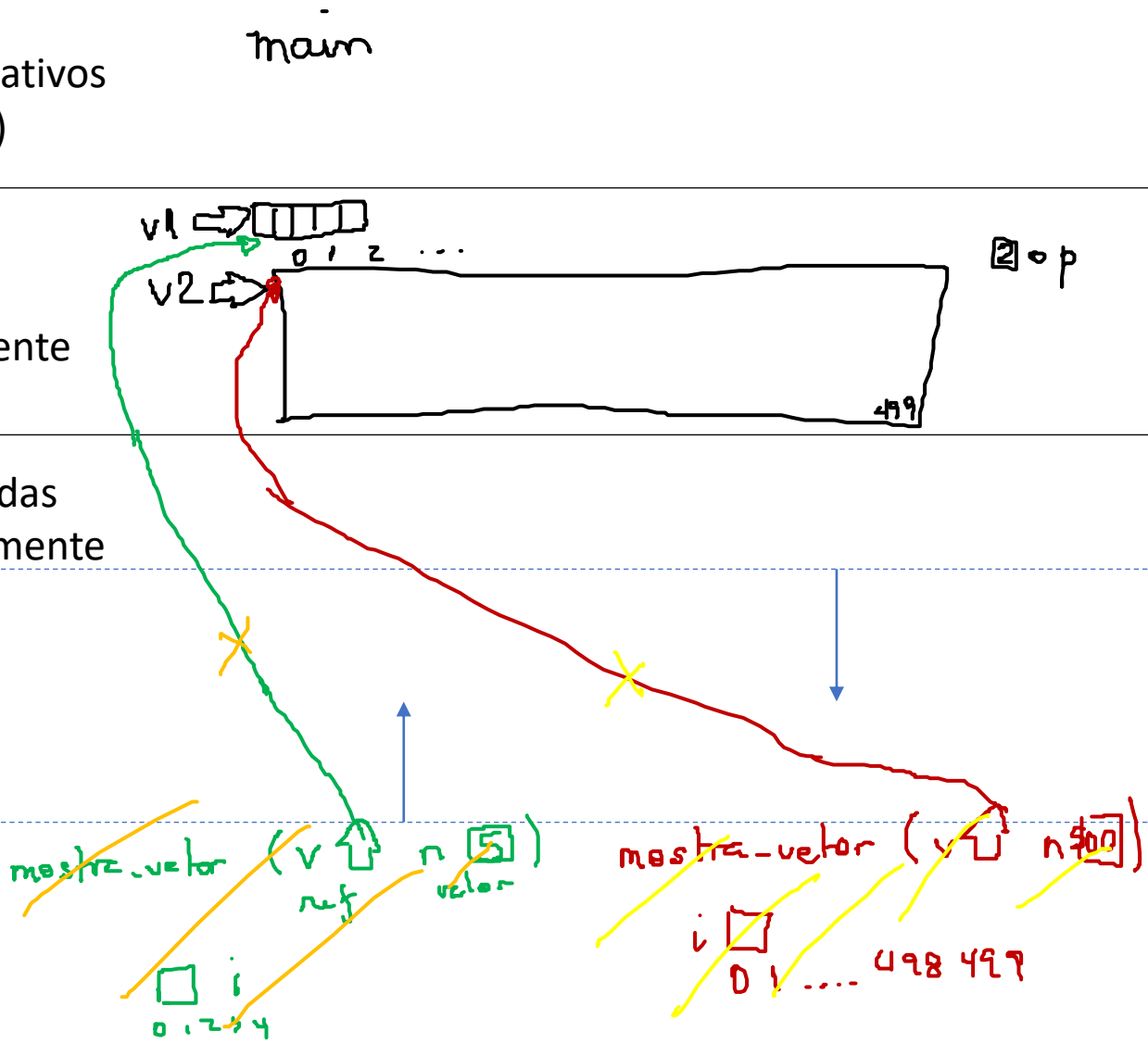
SO + aplicativos (principal)

Variáveis alocadas estaticamente + globais

Var. alocadas dinamicamente

livre

Pilha de execução




```
include <stdio.h>
oid le_vetor (int v[], int n) {
   int i;
   for (i=0; i<n; i++)
       scanf ("%d", &v[i]);

oid mostra_vetor (int v[], int n) {
   int i;
   for (i=0; i<n; i++)
       printf ("%d\n", v[i]);

nt main () {
   int v1[5], v2[500];
   int opcao;
   printf ("digite 1 para ler vetor ou 2 para
   scanf ("%d", &opcao);
   if (opcao == 1) {
       le_vetor(v1, 5);
       le_vetor(v2, 500);
   }
   else {
       mostra_vetor(v1, 5);
       mostra_vetor(v2, 500);
   }
   return 0;
```

endereco de v1 = 0061FF0C
v = 0061FF0C
n = 5
endereco de v2 = 0061F73C
v = 0061F73C
n = 500

sqrt

→ pow (x, 2)

→ x * x

protótipo de uma função?

→ declaração

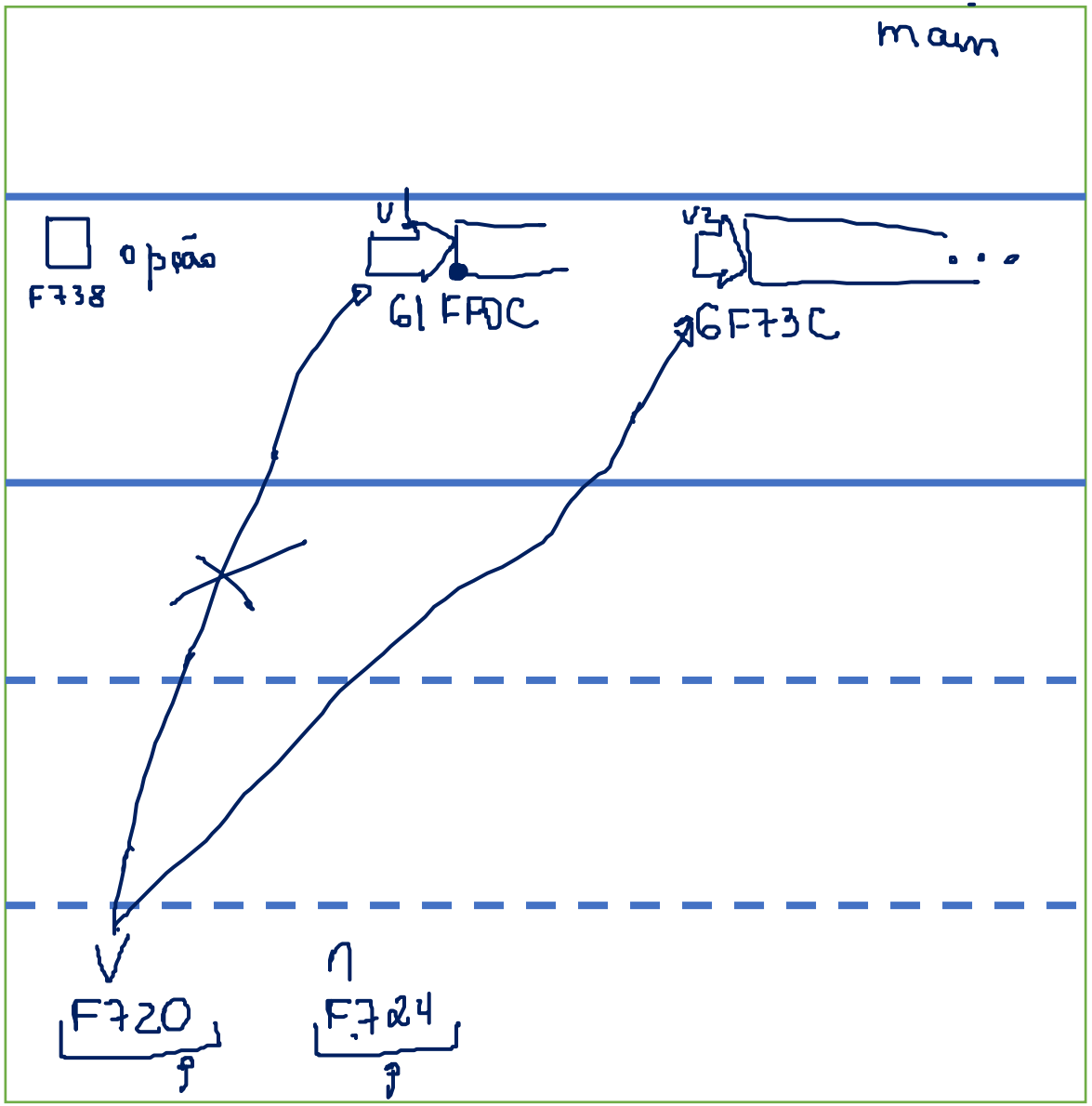
tipo_retorno identificador (parâmetros);

void le_vetor (int v[], int n);

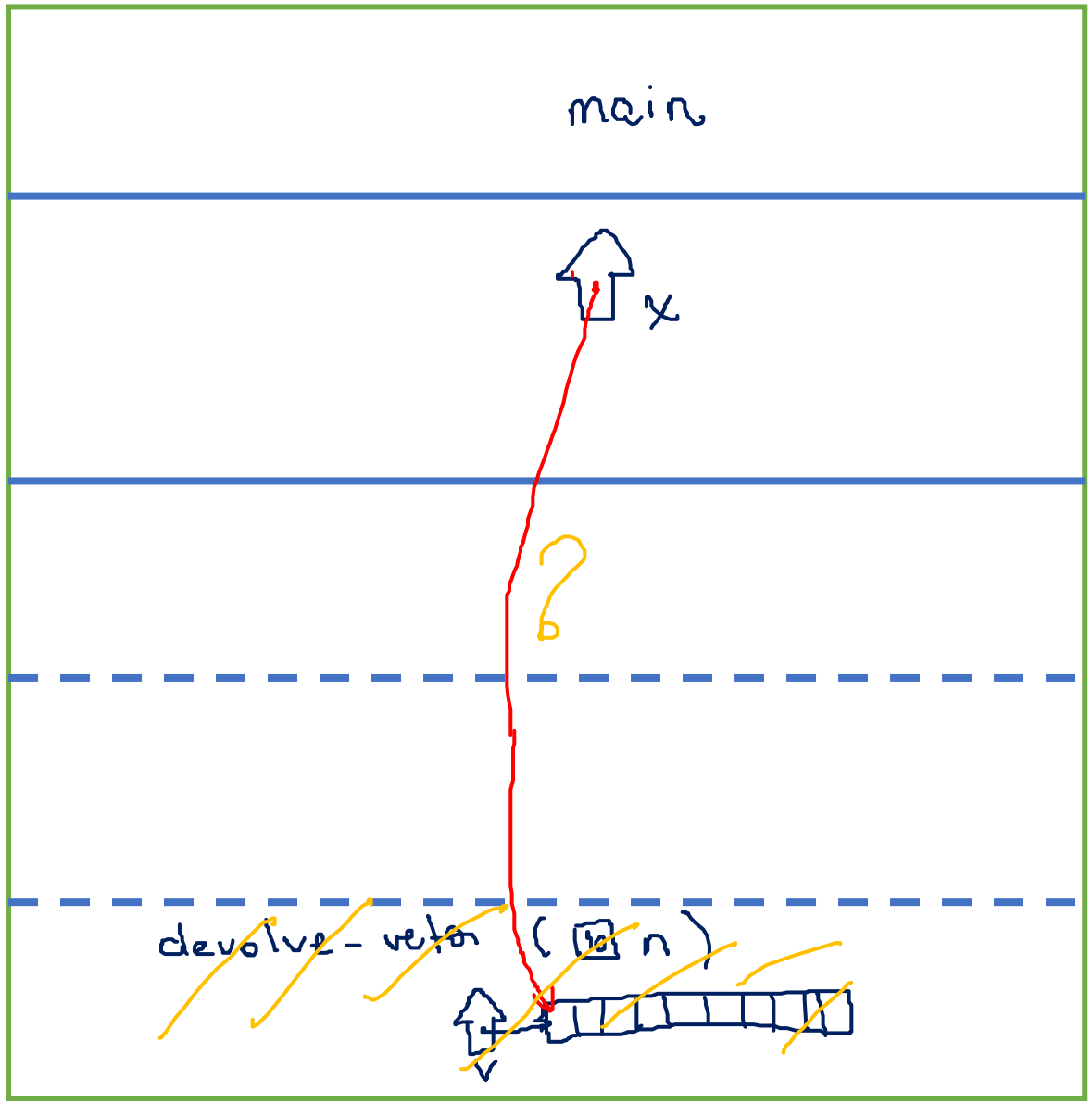
int *

↓

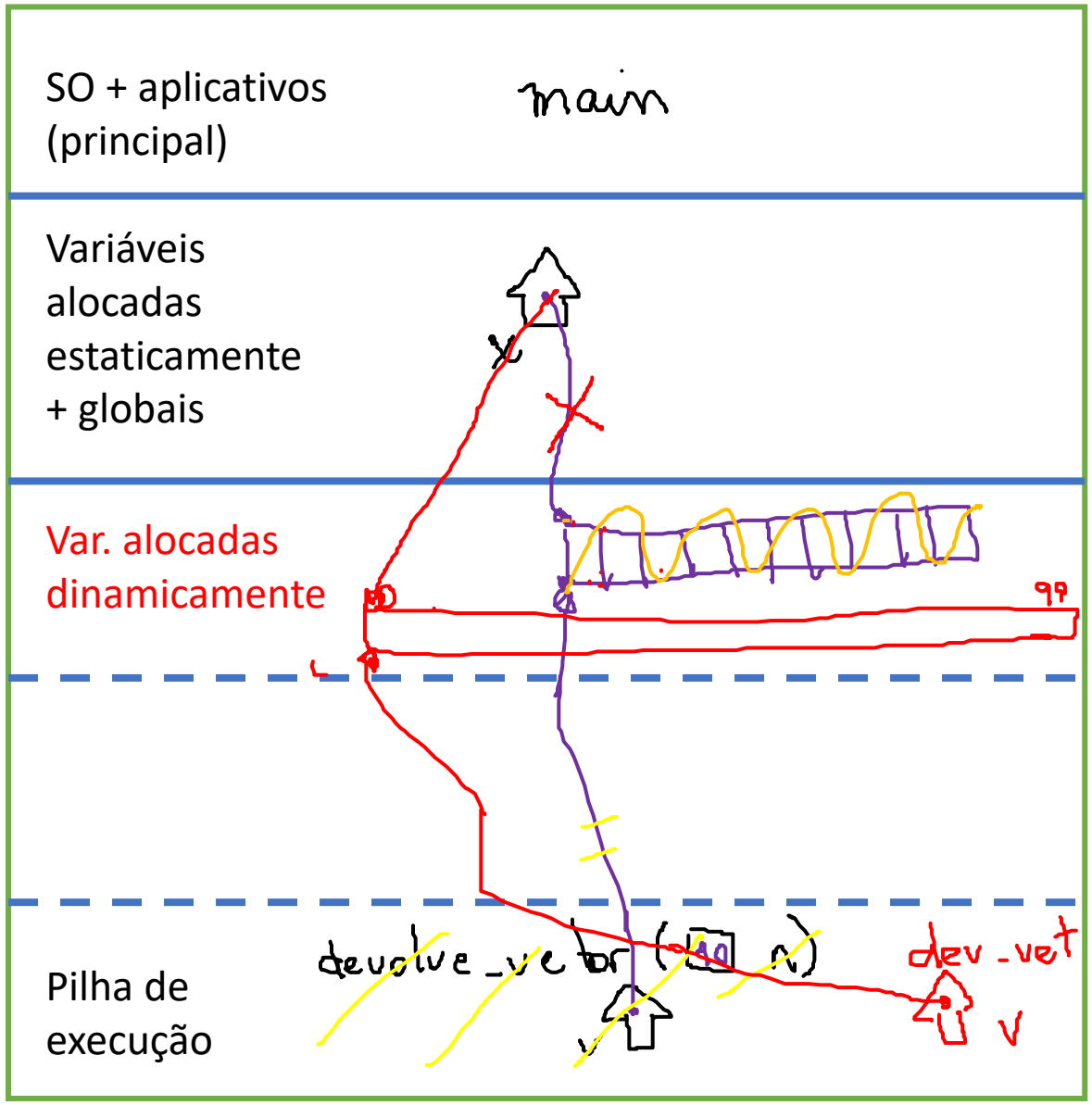
vetor inteiro



endereco da variavel opcao = 0061F738
digite 1 para ler vetor ou 2 para exibir vetor2
endereco do vetor v1 = 0061FF0C
endereco da variavel v1 = 0061FF0C
v = 0061FF0C
n = 5
endereco da variavel v = 0061F720
endereco da variavel n = 0061F724
endereco do vetor v2 = 0061F73C
endereco da variavel v2 = 0061F73C
v = 0061F73C
n = 500
endereco da variavel v = 0061F720
endereco da variavel n = 0061F724




```c
#include <stdio.h>

int * devolve_vetor (int);

int main () {
    int *x;
    x = devolve_vetor(10);
    return 0;
}

int * devolve_vetor (int n) {
    int v[n];
    return v;
}
```




```c
#include <stdio.h>
#include <stdlib.h>
int * devolve_vetor (int);

int main () {
    int *x;
    x = devolve_vetor(10);
    //processamento da lista de valores x
    free(x);
    x = devolve_vetor(100);
    //...
    free(x);
    return 0;
}
int * devolve_vetor (int n) {
    int *v = (int *) malloc (n * sizeof(int));
    return v;
}
```

Para ver se entendeu

1. Assinale a alternativa correta com relação ao estudo de Ponteiros?

- ○ Ponteiro é o valor de uma variável
- ○ Ponteiro é o indicador da próxima variável a ser passada
- ○ Ponteiro é uma variável que armazena endereço
- ○ Ponteiro é o endereço que aponta para uma variável

2. Quais das seguintes instruções declaram um ponteiro para uma variável float?

- ○ float *p;
- ○ float p;
- ○ *float p;
- ○ float p*;

Continuando

1. Verifique se há erros no código a seguir, em caso afirmativos, como seria a correção.

```
int main() {
    int x, *p;
    x = 100;
    p = x;
    printf(“Valor de p: %d.\n”, *p);
    return 0;
}
```

2. Suponha que os elementos de um vetor v são do tipo int e cada int ocupa 8 bytes no seu computador. Se o endereço de v[0] é 55000, qual o valor da expressão v + 4?

```c
int a=5, b=12, c;
int *p;
int *q;
p = &a;
q = &b;
c = *p + *q;
printf("c = %d", c);
```

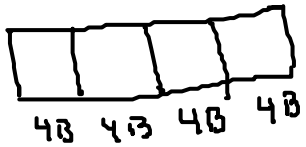
Seja o trecho de código acima, que valor de c será impresso no comando printf?

- c = 5
- c = 17
- c = 12
- c = 7

arrays : estruturas compostas homogêneas

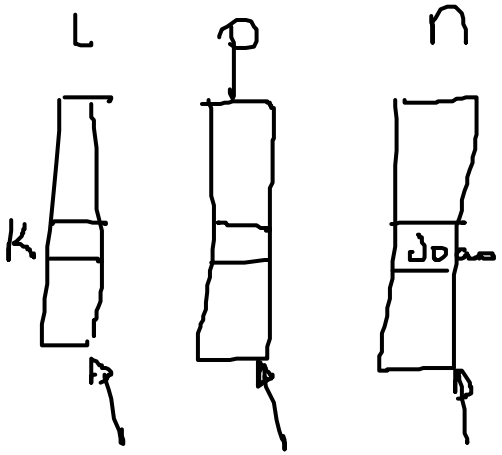
→ todos do mesmo tipo

→ vários valores

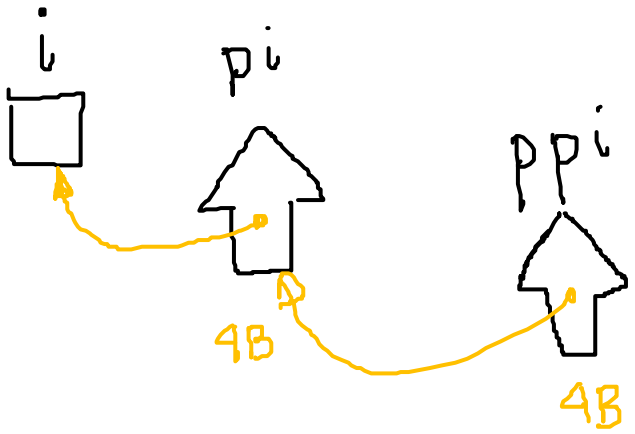


structs : estruturas compostas heterogêneas

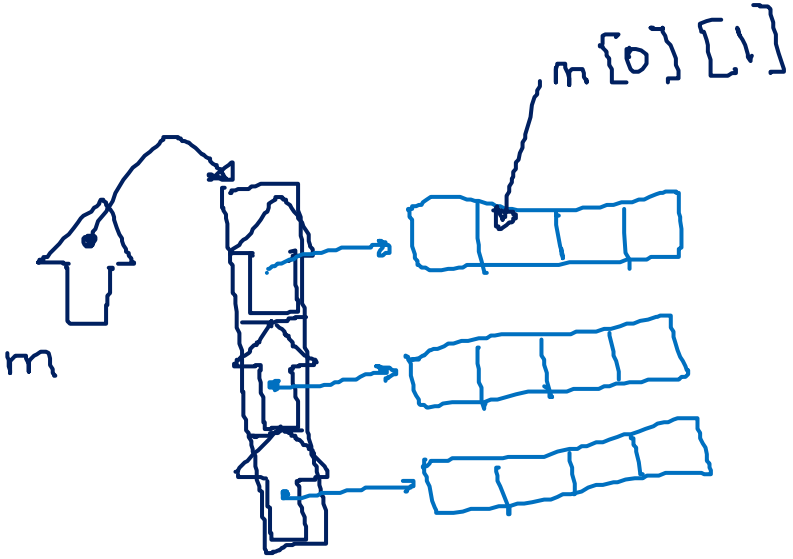
→ vários valores de tipos diferentes


L
K
A
D
n

```c
int main () {
    int i;
    int * pi;
    int ** ppi;

    int ** m;
    int m1[3][4];

    pi = &i;
    ppi = &pi;

    m = (int **) malloc (3 * sizeof(int *));
    for (i=0; i<3; i++) {
        m[i] = (int *) malloc (4 * sizeof(int));
    }
    return 0;
}
```

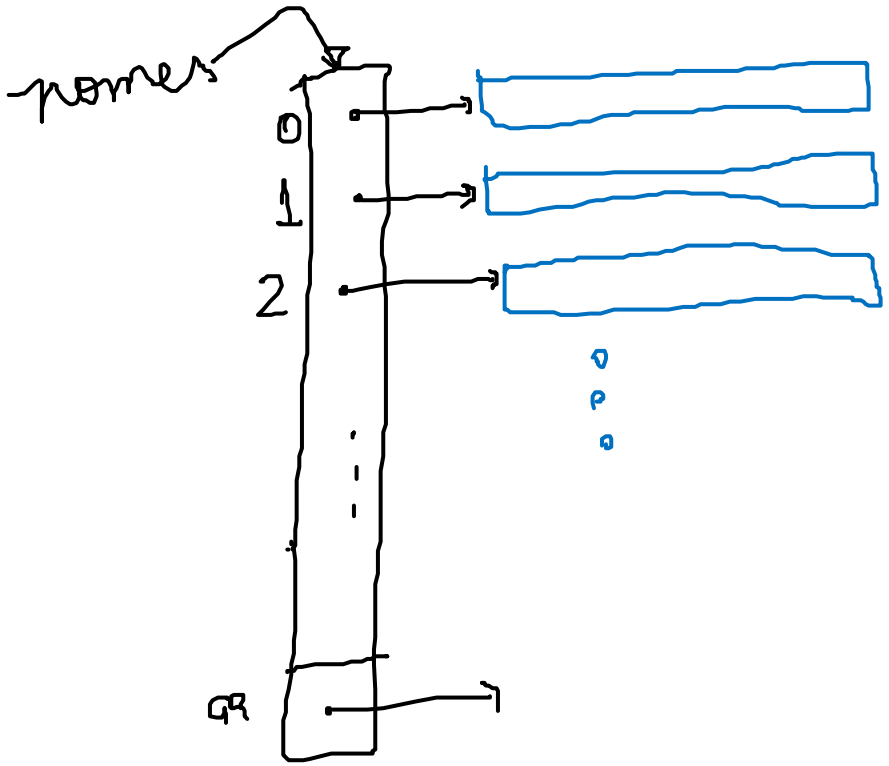


61789400

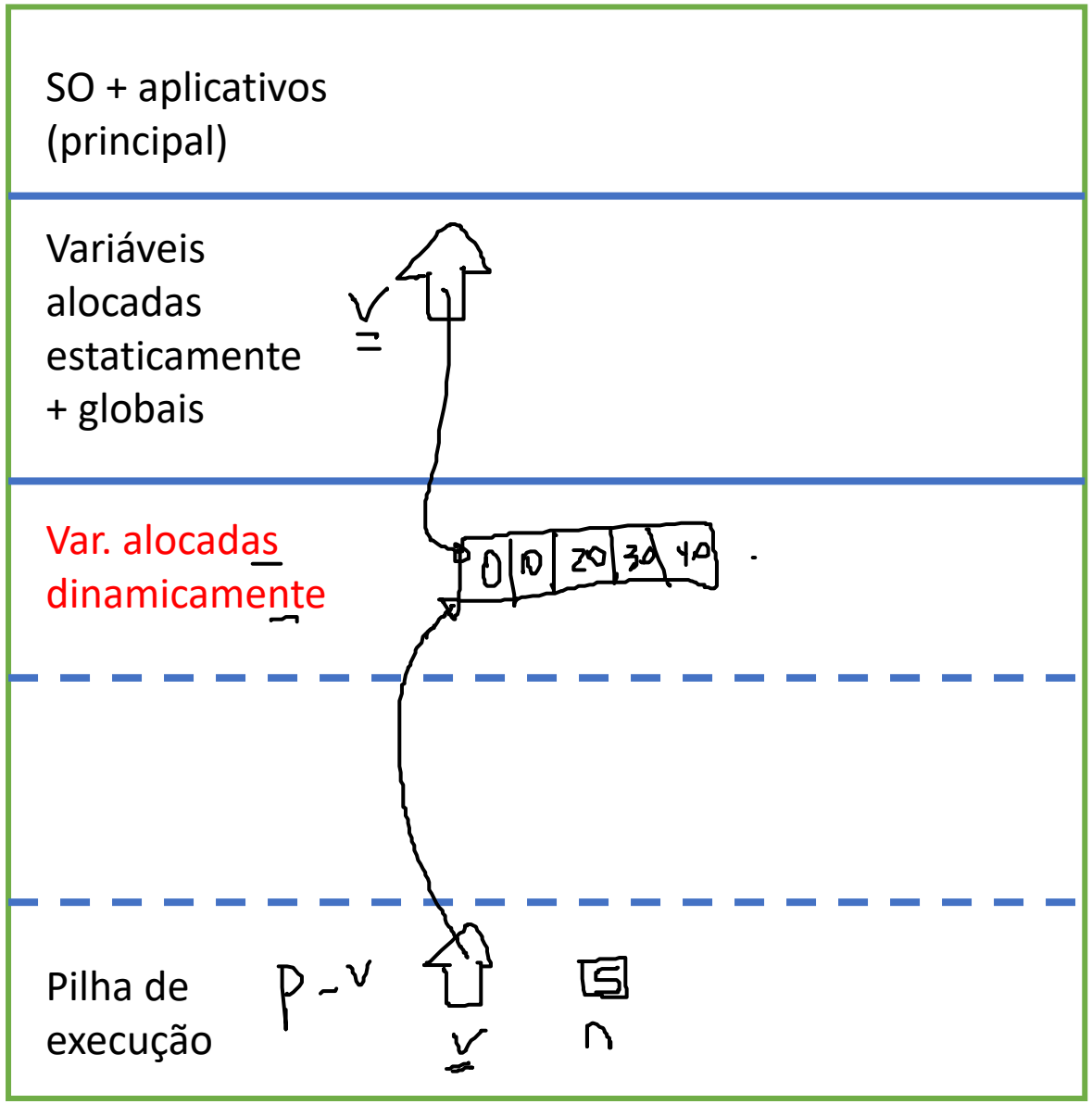



```
for(i=0; i<3; i++){
    for(j=0; j<4; j++){
        scanf("%d", &m[i][j]);}}
```

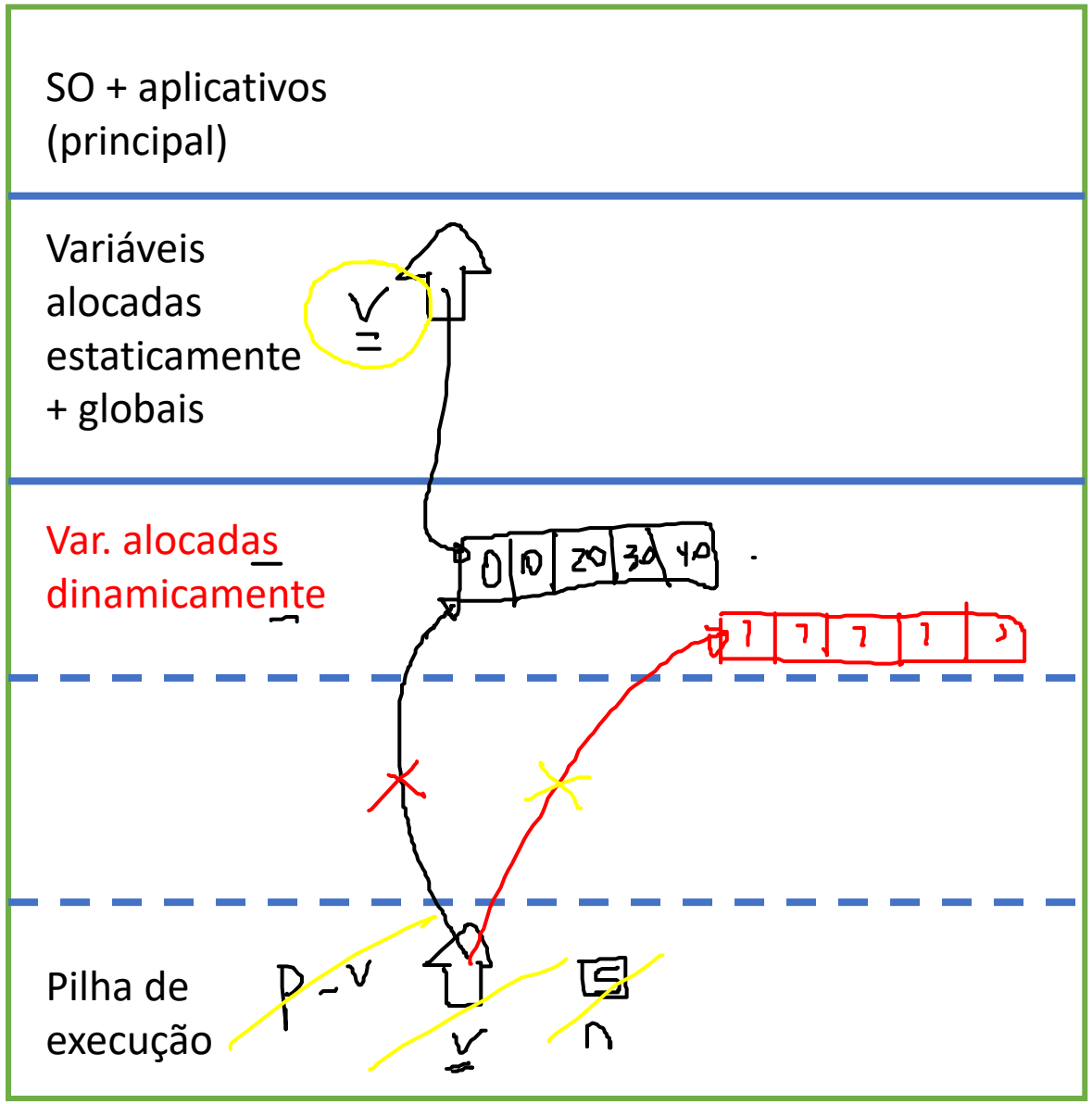
```
char ** nomes;

nomes = (char **) malloc (100 * sizeof (char *));
```



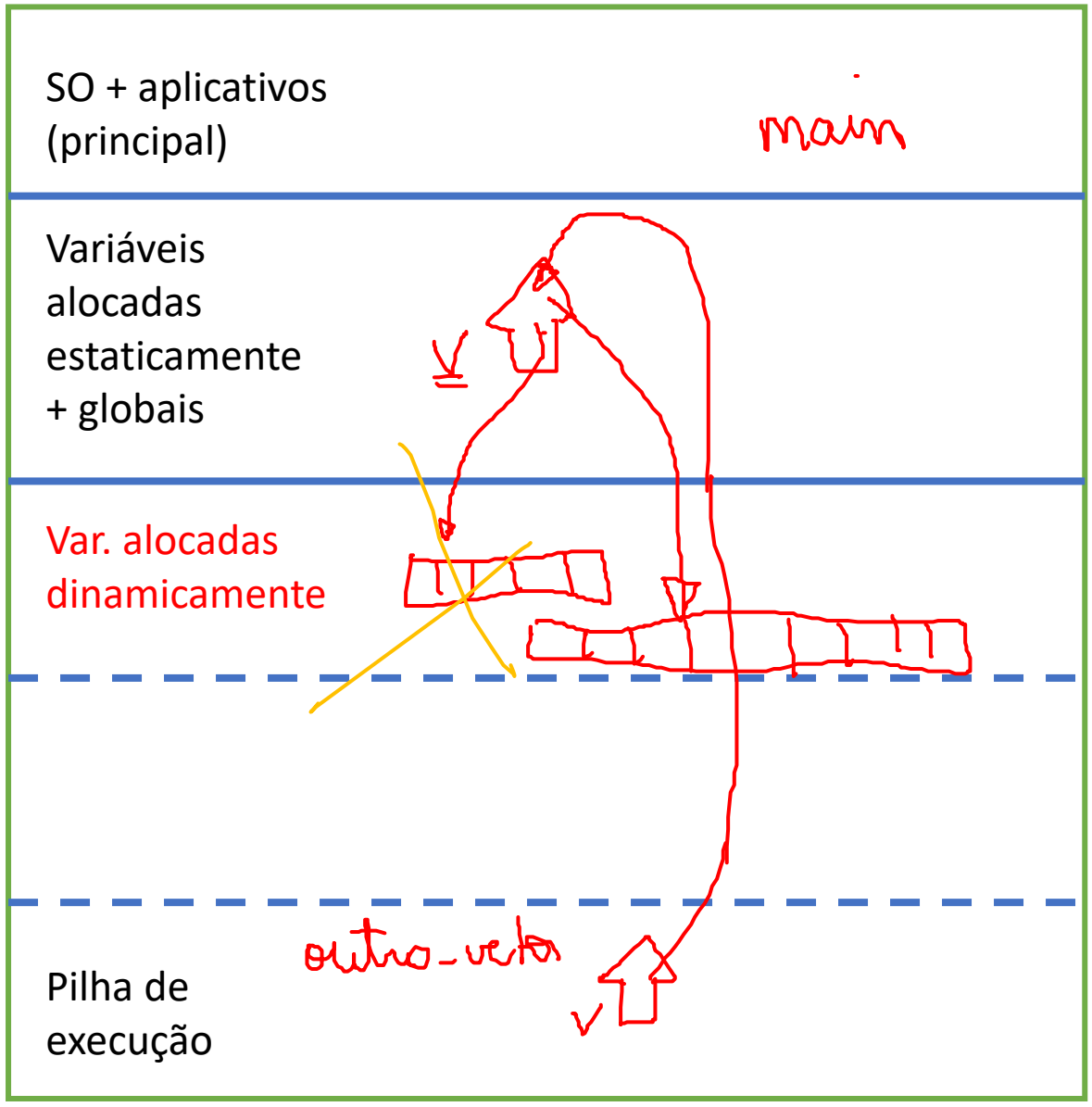

```
for (i=0; i<100; i++) {
    nomes[i] = (char *) malloc (40);
    scanf ("%s", nomes[i]);
}
```




```
void preenche_vetor (int *v, int n) {
    int i;
    for (i=0; i<n; i++) {
        v[i] = i*10;
    }
}

int main () {
    int *v = (int *) malloc (5 * sizeof(int));
    preenche_vetor (v, 5);

    return 0;
}
```




```c
#include <stdio.h>
#include <stdlib.h>

void preenche_vetor (int *v, int n) {
    int i;
    for (i=0; i<n; i++) {
        v[i] = i*10;
    }

    v = (int *) malloc (5 * sizeof (int));
}

int main () {
    int *v = (int *) malloc (5 * sizeof(int));
    preenche_vetor (v, 5);

    return 0;
}
```




```c
#include <stdio.h>
#include <stdlib.h>

void preenche_vetor (int *v, int n) {
    int i;
    for (i=0; i<n; i++) {
        v[i] = i*10;
    }
    v = (int *) malloc (5 * sizeof (int));
}
void outro_vetor (int **v, int n);
    *v = (int *) malloc (n * sizeof(int));
}
int main () {
    int *v = (int *) malloc (5 * sizeof(int));
    //preenche_vetor (v, 5);
    outro_vetor (&v, 10);
    return 0;
}
```

Agora vamos programar um pouco

1. Implemente uma função que recebe um número real passado valor, retorne a parte inteira e a parte fracionaria desse número. Depois, implemente a função main que chama essa função.

   Protótipo: void partes (float num, int *inteiro, float *fracionária);

2. Implemente uma função que calcula a área da superfície e o volume de uma esfera de raio r. Essa função deve obedecer ao protótipo:

   void calc_esfera (float r, float *area, float *volume);

   A área da superfície e o volume são dados, respectivamente, por:

   $$A = 4 * p * R^2$$

   $$V = 4/3 * p * R^3$$

3. Implemente uma função que recebe como parâmetro um array de inteiros com n valores e determina o maior elemento do array e o número de vezes que esse elemento ocorreu no array. Por exemplo, para o array {5, 2, 15, 3, 7, 15, 8, 6, 15}, a função deve retornar para a função que a chamou o valor 15 e o número 3, indicando que o número 15 ocorreu 3 vezes. A função deve ser do tipo void.