B.14 Formative Evaluation: Multiple Choice Knowledge Questionnaires of Loop Structures (provinha2a)

CATALOG

SSC0600 - Introdução à Ciência de Computação I **Provinha 2(a)**
Tópico: Estruturas de Repetição e Tipos de Dados Compostos (Strings e Vetores)
25 de abril de 2017

N.º USP:

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

← Por favor codifique seu Número USP na esquerda e escreva seu nome abaixo.

Nome e sobrenome:

.....

.....

Question [`remember-unistructural`] ♣ Marque (X) nas opções que são usadas para representar as **estruturas de repetição** (*loops*) em pseudocódigo ou na Linguagem C.

- ■ while (*expressão*) { ... }
- ■ for (*expressão*; *expressão*; *expressão*) { ... }
- ■ do { ... } while (*expressão*);
- ☐ if (*expressão*) { ... }
- ☐ if (*expressão*) { ... } else { ... }
- ☐ if-not (*expressão*) { ... }
- ☐ if-not (*expressão*) { ... } else { ... }
- ☐ while (*expressão*) { ... } other-case { ... }
- ☐ do { ... } until (*expressão*) { ... }
- ■ PARA *expressão* ATÉ *expressão* PASSO *expressão* FAÇA ... FIMPARA
- ■ ENQUANTO *expressão* FAÇA ... FIMENQUANTO
- ■ REPITA *expressão* ATÉ *expressão*
- ☐ SE *expressão* ENTÃO ...
- ☐ SE *expressão* ENTÃO ... SENÃO ...
- ☐ SENÃO *expressão* ENTÃO ...
- ☐ SENÃO *expressão* ENTÃO ... SENÃO ...
- ☐ PARA *expressão* ATÉ *expressão* FAÇA ... FIMPARA
- ☐ ENQUANTO *expressão* FAÇA ... EMOUTROSCASOSFAÇA ... FIMENQUANTO
- ☐ *Nenhuma das alternativas está correta*

CATALOG

Question [`understand-relational`] ♣ Marque (X) nas afirmativas verdadeiras em relação aos trechos de código em Linguagem C apresentados nas Listagem 1, 2, 3 e 4

- *Observação*: `arr[n][n]` é a matriz n*n de números inteiros

- [x] Listagem 1 gira 180 graus no sentido horário ou antihorário a matriz `arr[n][n]`
- [x] Listagem 2 gira 90 graus no sentido horário a matriz `arr[n][n]`
- [x] Listagem 3 gira 360 graus no sentido horário ou antihorário a matriz `arr[n][n]`
- [x] Listagem 4 gira 90 graus no sentido anti-horário a matriz `arr[n][n]`
- [ ] Listagem 1 gira 360 graus no sentido horário ou antihorário a matriz `arr[n][n]`
- [ ] Listagem 2 gira 270 graus no sentido horário a matriz `arr[n][n]`
- [ ] Listagem 4 gira 270 graus no sentido anti-horário a matriz `arr[n][n]`
- [ ] Listagem 2 gira 90 graus no sentido anti-horário a matriz `arr[n][n]`
- [ ] Listagem 4 gira 90 graus no sentido horário a matriz `arr[n][n]`
- [ ] Listagem 1 gira 90 graus no sentido horário a matriz `arr[n][n]`
- [ ] Listagem 2 gira 360 graus no sentido horário ou antihorário a matriz `arr[n][n]`
- [ ] Listagem 3 gira 90 graus no sentido anti-horário a matriz `arr[n][n]`
- [ ] Listagem 4 gira 360 graus no sentido horário ou antihorário a matriz `arr[n][n]`
- [ ] *Nenhuma das alternativas está correta*

Question [`apply-multistructural-1`] ♣ Marque (X) na saída do programa apresentado na Listagem 6.

- [x]
  ```
  0 1 2
  0 1 2
  0 1 2
  ```
- [ ]
  ```
  0 0 0
  1 1 1
  2 2 2
  ```
- [ ]
  ```
  2 1 0
  2 1 0
  2 1 0
  ```
- [ ]
  ```
  2 2 2
  2 2 2
  2 2 2
  ```
- [ ] *Nenhuma das alternativas está correta*

Question [`apply-multistructural-2`] ♣ Marque (X) nas afirmativas verdadeiras em relação ao programa da Listagem 7.

- [x] Depois que o código for executado, x contém os valores: {5, 4, 3, 4, 2, 0}
- [ ] Depois que o código for executado, x contém os valores: {5, 4, 2, 3, 2, 0}
- [ ] Depois que o código for executado, x contém os valores: {10, 8, 6, 2, 1, 0}
- [ ] Depois que o código for executado, x contém os valores: {10, 8, 2, 3, 1, 0}
- [x] Depois que o código for executado, y contém os valores: {5, 2, 6, 4, 4, 0}
- [ ] Depois que o código for executado, y contém os valores: {5, 2, 4, 3, 4, 0}
- [ ] Depois que o código for executado, y contém os valores: {5, 1, 6, 4, 8, 0}
- [ ] Depois que o código for executado, y contém os valores: {5, 1, 2, 6, 8, 0}
- [ ] *Nenhuma das alternativas está correta*

CATALOG

Question [evaluate-multistructural] ♣ Marque (X) nas afirmativas verdadeiras em relação ao programa da Listagem 7.

- [x] A condição `i1 < j1` da Linha 9 é avaliada 4 veces
- [ ] A condição `i1 < j1` da Linha 9 é avaliada 3 veces
- [ ] A condição `i1 < j1` da Linha 9 é avaliada 6 veces
- [x] A condição `j2 > j2` da Linha 16 é avaliada 4 veces
- [ ] A condição `j2 > j2` da Linha 16 é avaliada 3 veces
- [ ] A condição `j2 > j2` da Linha 16 é avaliada 6 veces
- [x] O código na estrutura de repetição externa (*outer loop*), linhas 10 até 25, é repetido 3 veces
- [ ] O código na estrutura de repetição externa (*outer loop*), linhas 10 até 25, é repetido 4 veces
- [ ] O código na estrutura de repetição externa (*outer loop*), linhas 10 até 25, é repetido 6 veces
- [x] O código na estrutura de repetição interna (*inner loop*), linhas 17 até 21, é repetido 3 veces
- [ ] O código na estrutura de repetição interna (*inner loop*), linhas 17 até 21, é repetido 4 veces
- [ ] O código na estrutura de repetição interna (*inner loop*), linhas 17 até 21, é repetido 6 veces
- [ ] *Nenhuma das alternativas está correta*

Question [analyse-relational-1] ♣ Marque (X) nas modificações que, de maneira independente umas das outras, façam com que o programa apresentado na Listagem 6 imprima:

0 0 0
1 1 1
2 2 2

- [x] A linha 14 deve ser mudada para: `printf("%d ", m[j][i]);`
- [x] A linha 9 deve ser mudada para: `m[row][col] = row;`
- [x] A linha 9 deve ser mudada para: `m[col][row] = col;`
- [ ] A linha 12 deve ser mudada para: `for (i=2; i>=0; i--) {`
- [ ] A linha 13 deve ser mudada para: `for (j=2; j>=0; j--) {`
- [x] A linha 13 deve ser mudada para: `for (j=2; j>=0; j--) {`
  A linha 14 deve ser mudada para: `printf("%d ", m[j][i]);`
- [ ] A linha 12 deve ser mudada para: `for (i=2; i>=0; i--) {`
  A linha 14 deve ser mudada para: `printf("%d ", m[j][i]);`
- [ ] *Nenhuma das alternativas está correta*

Question [analyse-relational-2] ♣ O trecho de código apresentado na Listagem 5 tem sido proposto para efetuar a ordenação ascendente (de menor a maior) de um vetor de inteiros `arr[n]` (array `arr` de tamanho n). No entanto, o programa não funciona adequadamente. Marque (X) nas modificações necessárias que, em conjunto, façam o programa funcionar adequadamente.

- [x] A linha 5 deve ser mudada para: `if (arr[j] < arr[m]) {`
- [ ] A linha 5 deve ser mudada para: `if (arr[j] > arr[m]) {`
- [ ] A linha 5 deve ser mudada para: `if (arr[j] >= arr[m]) {`
- [x] A linha 4 deve ser mudada para: `while (j < n) {`
- [ ] A linha 4 deve ser mudada para: `while (j < n-1) {`
- [ ] A linha 4 deve ser mudada para: `while (j > n) {`
- [ ] A linha 4 deve ser mudada para: `while (j >= n-1) {`
- [ ] *Nenhuma das alternativas está correta*

```
for(j=n-1; j>=0; j--) {
    for(k=n-1; k>=0; k--) {
        printf("%d ", arr[j][k]);
    }
    printf("\n");
}
```

Listagem 1: Trecho de código para girar uma matriz n*n

```
for (j=0; j<n; j++) {
    for(k=n-1; k>=0; k--) {
        printf("%d ", arr[k][j]);
    }
    printf("\n");
}
```

Listagem 2: Trecho de código para girar uma matriz n*n

```
for(j=0; j<n; j++) {
    for(k=0; k<n; k++){
        printf("%d ", arr[j][k]);
    }
    printf("\n");
}
```

Listagem 3: Trecho de código para girar uma matriz n*n

```
for(j=n-1; j>=0; j--) {
    for(k=0; k<n; k++){
        printf("%d ", arr[k][j]);
    }
    printf("\n");
}
```

Listagem 4: Trecho de código para girar uma matriz n*n

```
for (i = 0; i < n-1; i++) {
    m = i;
    j = i+1;
    while (n-1 > j) {
        if (arr[m] < arr[j]) {
            m = j;
        }
        j++;
    }

    if (i != m) {
        aux = arr[i];
        arr[i] = arr[m];
        arr[m] = aux;
    }
}
```

Listagem 5: Trecho de código para ordenar um vetor de inteiros `arr[n]`

```c
#include <stdio.h>

int row, col, i, j;
int m[3][3];

int main() {
    for (row = 0; row <= 2; row++) {
        for (col = 0; col <= 2; col++) {
            m[row][col] = col;
        }
    }
    for (i=0; i<3; i++) {
        for (j=0; j<3; j++) {
            printf("%d ", m[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Listagem 6: Código de programa na linguagem C

```c
#include <stdio.h>

int x[6] = {0, 1, 2, 3, 4, 5};
int y[6] = {5, 4, 3, 2, 1, 0};
int i1 = 0, j1 = 5, i2 = 0, j2 = 5;
int temp;

int main() {
    while (i1 < j1) {
        temp = x[i1];
        x[i1] = x[j1];
        x[j1] = 2*temp;

        i2 = i1+1;
        j2 = j1-1;
        while (j2 > i2) {
            temp = y[j2];
            y[j2] = y[i2];
            y[i2] = 2*temp;
            i2++;
            j2--;
        }

        i1++;
        j1--;
    }
    return 0;
}
```

Listagem 7: Código de programa na linguagem C