Apontadores

1. Considere as seguintes declarações de variáveis:

int W[5], **V, M, k;

e que **sizeof(int) = 4** e **sizeof(int*) = 8**.

a) Preencha o esquema dado à esquerda considerando a execução do bloco de instruções em linguagem C dado à direita.

(esquema de um bloco de memória)

	. . .	
112000	**114000**	**V**
	. . .	
113000		**M**
	. . .	
114000	**114190**	
	. . .	
114070		
114074		
114078		
114082		
114086		
114190		
114094		
	. . .	
115000	**114078**	**W**
	. . .	

Bloco de instruções em linguagem C:

```
M = 5;
for (k = 1; k < M; k++)
    W[k] = 5*k + 10;
*W = 40;
*(W-1) = 50;
```

b) Com os valores obtidos com a execução do bloco de instruções dado, determine os valores das seguintes expressões:

EXPRESSÃO	VALOR
*W - 2	
W - 2	
&W[2]	
&(*W)	
**V - 2	
*(*V - 2)	
V - 2	

2. Considere as seguintes declarações de variáveis:

int **V, W[8], N, B;

Considere também que **sizeof(int) = 4** e **sizeof(int*) = 8**.

a) Preencha o esquema dado à esquerda considerando a execução do bloco de instruções em linguagem C dado à direita.

(esquema de um bloco de memória)

	. . .	
100500	**110160**	**W**
	. . .	
100700	**110172**	
	. . .	
100800		**N**
	. . .	
110160	**-100**	
110164		
110168		
110172		
110176		
110180		
110184		
110188		
110192	**-200**	**B**
	. . .	
110500	**100700**	**V**
	. . .	

Bloco de instruções em linguagem C:

```
**V = 900;
for (N = 2; N < 5; N++)
   (*V)[N-1] = N * 100;
*(W + 1) = 600;
*(*V - 1) = 700;
W[7] = 100;
```

b) Com os valores obtidos com a execução do bloco de instruções dado, determine os valores das seguintes expressões:

EXPRESSÃO	VALOR
**V + 3	
*V + 3	
V + 3	
(*V)[4]	
*(*V - 3)	
**V + 4	
&(*V)[2]	
W - 3	
W[3]	
&(W[2])	
W[8]	
*(W + 5)	
*W - 2	

3. Considere as seguintes declarações de variáveis:

int V[7], **W, N;

Considere também que **sizeof(int) = 4** e **sizeof(int*) = 8**.

a) Preencha o esquema dado à esquerda, considerando a execução do bloco de instruções em linguagem C dado à direita.

(esquema de um bloco de memória)

	. . .	
100500	**110180**	
	. . .	
100700	**110164**	**V**
	. . .	
100800		**N**
	. . .	
110160	**100**	
110164		
110168		
110172		
110176		
110180		
110184		
110188		
110192	**200**	
	. . .	
110500	**100500**	**W**
	. . .	

Bloco de instruções em linguagem C:

```
*V = 900;
N = 1;
do{
   V[N] = N * 500;
   N = N + 1;
}while (N < 4);
*(*W + 1) = 600;
*(V + 6) = 400;
(*W)[0] = 50;
```

b) Usando os valores obtidos com a execução do bloco de instruções dado, determine os valores das seguintes expressões:

EXPRESSÃO	VALOR
V + 4	
*(V + 4)	
*V + 4	
V[4]	
&V[4]	
&(*V + 4)	
*V - 4	
**W	
**W - 2	
(*W)[2]	
&(*W)[2]	
*(*W + 3)	
W - 2	

4. Um ponteiro pode ser manipulado como sendo um array de 1 dimensão. Considere o seguinte programa em linguagemC:

```
void main ()
{
   int  v[5] = { 10, 20, 30, 40, 50 };
   int  p, i;
   p = v;
   for (i = 1; i < 5; i++)
      printf("%d ", p[i]);
}
```

a) Escreva o programa anterior no computador, identifique e corrija os erros no código.

b) Acrescente ao programa anterior uma instrução para escrever o endereço do 1º elemento do array, usando as seguintes 4 formas diferentes para o fazer: &v[0], &p[0], v, p.

c) Acrescente ao programa anterior uma instrução para escrever o endereço do 2º elemento do array, usando as seguintes 4 formas diferentes para o fazer: &v[1], &p[1], v+1, p+1. Quantos bytes ocupa cada valor do array?

d) Acrescente ao programa anterior uma instrução para escrever o valor do 1º elemento do array, usando as seguintes 4 formas diferentes para o fazer: v[0], p[0], *v, *p.

e) Acrescente ao programa anterior uma instrução para escrever o valor do 2º elemento do array, usando as seguintes 4 formas diferentes para o fazer: v[1], p[1], *(v+1), *(p+1).

f) Acrescente ao programa anterior a instrução que se segue e compare os resultados com os obtidos na alínea anterior:

```
printf("%d %d %d %d\n", v[1], p[1], *v+1, *p+1);
```

5. Indicar e justificar (através de simulação) o que faz o seguinte programa:

```
void main ()
{
   int  y, *p, x;
   y = 0;
   p = &y;
   x = *p;
   x = 4;
   (*p) = (*p) + 1;
   x = x - 1;
   (*p) = (*p) + x;
   printf("y = %d\n", y);
}
```

6. Indicar e justificar (através de simulação) o que faz o seguinte programa:

```
void main ()
{
   int   *pont, cont, val;
   cont = 100;
   pont = &cont;
   val = *pont;
   printf("%d", val);
}
```

7. Indicar e justificar (através de simulação) o que faz o seguinte programa:

```
void main ()
{
   int   x, y, *px, *py;
   x = 5;
   px = &x;
   py = px;
   y = *py;
   printf("%d %d", x, y);
}
```

8. Indicar e justificar (através de simulação) o que faz o seguinte programa:

```
void main ()
{
   int   x, y, *px, **py;
   x = 5;
   px = &x;
   py = &px;
   y = **py;
   printf("%d %d", x, y);
}
```

9. Indicar e justificar (através de simulação) o que faz o seguinte programa:

```
void main ()
{
   char   a, b, *p;
   b = 'c';
   p = &a;
   *p = b;
   printf("%c", a);
}
```

10. Indicar e justificar (através de simulação) o que faz o seguinte programa:

```
void main ()
{
   int   x, y, *px, *py;
   printf("Digite um valor: ");
   scanf("%d", &x);
   px = &x;
   y = *px;
   printf("digitou = %d e y = %d\n", x, y);
   *px = 8;
   printf("valor mudou para %d\n", x);
}
```

11. Indicar e justificar (através de simulação) o que faz o seguinte programa:

```
void main ()
{
   int  i, k, *pi, *pk;
   char  a;
   i = 2;
   k = 0;
   printf("Qual será o valor de k? ");
   pk = &k;
   pi = &i;
   *pk = i;
   printf("para *pk = I, temos k = %d\n", k);
   k = *pi;
   printf("para k = *pi, temos k = %d\n", k);
   scanf("%c", &a);
}
```

12. Escreva um programa que some dois valores inteiros. Para tal, deverá implementar e utilizar os subprogramas seguintes:

a) int soma1 (int a, int b)

b) int soma2 (int *a, int *b)

13. Verificar que o subprograma **soma3**, implementado da forma que segue, está incorreta. Explicar o comportamento indesejado utilizando, se necessário, exemplos de execução.

```
int  *soma3 (int a, int b)
{
   int  temp ;
   temp = a+b;
   return (&temp);
}
```

14. Escreva um programa que calcule o maior elemento, e o respetivo índice/posição, de um array 1D com no máximo 100 números reais. Para tal, use os subprogramas da biblioteca "**Array1DReais**", que se encontra disponível na página web da disciplina, e acrescente à biblioteca um outro subprograma com o seguinte protótipo:

int maior_indicemaior (float[], int, float*);

15. Escreva um programa que calcule a soma e a média de um um array 1D com no máximo 100 números reais. Para tal, use os subprogramas da biblioteca "**Array1DReais**", que se encontra disponível na página web da disciplina, e acrescenta à biblioteca um outro subprograma com o seguinte protótipo:

void soma_media (float[], int, float*, float*);

16. Construa um programa que:

- leia um array 1D com no máximo 100 números reais
- crie um segundo array Y apenas com os números positivos do array X
- mostre no monitor todos os elementos dos arrays X e Y

Para tal, use os subprogramas da biblioteca "**Array1DReais**" (disponível na página web da disciplina), e acrescente à biblioteca um outro subprograma com o seguinte protótipo:

void arrayPositivos (float[], int, float[], int*);

17. Escreva um programa que leia **números reais**, guarde-os num **array 2D**, em que o número máximo de linhas e o número máximo de colunas são valores especificados pelas constantes **MAXLIN** e **MAXCOL**, e determine o maior elemento e o menor elemento do **array**. Para tal, acrescente à biblioteca "Array2DReais" um subprograma com o seguinte protótipo:

void maiorMenorArray2DReais (float[][], int, int, float*, float*);

18. Escreva um programa que leia **números reais**, guarde-os num **array 2D**, em que o número máximo de linhas e o número máximo de colunas são valores especificados pelas constantes **MAXLIN** e **MAXCOL**, e determine os índices (linha e coluna) do maior elemento e do menor elemento do **array**. Para tal, acrescente à biblioteca "Array2DReais" dois subprogramas com os seguintes protótipos:

void indicesMaiorArray2DReais (float[][], int, int, int*, int*);

void indicesMenorArray2DReais (float[][], int, int, int*, int*);