



Universidade Federal de São João del Rei
Departamento de Ciência da Computação
Curso de Ciência da Computação

Roteiro 1

Rodrigo José Zonzin Esteves
212050002

1 Ponteiros

1.1

Código

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4
5 int negativos(int*, int);
6
7 int negativos(int *vet, int N){
8     int count = 0;
9
10    for(int i = 0; i<N; i++){
11        if(vet[i] < 0) count++;
12    }
13
14    return count;
15 }
16
17
18 int main(){
19     int vet[5] = {-1, 2, -3, -4, 5};
20     printf("%d\n", negativos(vet, sizeof(vet)/sizeof(vet[0])));
21
22
23     return 0;
24 }
```

../q1_l1.c

Saída

```
zonzin@rodrigo:~/Documentos/Faculdade/labaedsil/labaeds2/pr1$ ./q1
3
zonzin@rodrigo:~/Documentos/Faculdade/labaedsil/labaeds2/pr1$
```

Figura 1: Q1

1.2

Código

ATENÇÃO: para a execução correta desse código é necessário passar o tamanho desejado n como argumento da main.

./q2 10

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <limits.h>
4 #include <time.h>
5
6 int* preenche_vetor(int n);
7 int maior_vetor(int*, int);
8 int menor_vetor(int*, int);
9 float media_vetor(int*, int);
10 void desaloca(int*);
11
12 int main(int argc, char **argv){
13     int n = atoi(argv[1]);
14     int *v = preenche_vetor(n);
15
16
17     printf("Maior: %d\n",
18           maior_vetor(v, n));
19     printf("Menor: %d\n",
20           menor_vetor(v, n));
21     printf("Media: %.2f\n",
22           media_vetor(v, n));
23
24     desaloca(v);
25     return 0;
26 }
27
28 int maior_vetor(int *v, int n){
29     int maior = v[0];
30
31     for(int i = 0; i<n; i++){
32         if(v[i] >= maior){
33             maior = v[i];
34         }
35     }
36     return maior;
37
38 int menor_vetor(int *v, int n){
39     int menor = v[0];
40
41     for(int i = 0; i<n; i++){
42         if(v[i] <= menor){
43             menor = v[i];
44         }
45     }
46     return menor;
47 }
48
49 float media_vetor(int *v, int n){
50     int sum = 0;
51     float media;
52
53     for(int i = 0; i<n; i++){
54         sum += v[i];
55     }
56     return (float)sum/n;
57 }
58
59 int* preenche_vetor(int n){
60     int * vetor =
61         (int*)malloc(sizeof(int)*n);
62     srand((time_t)time(NULL));
63
64     for(int i = 0; i<n; i++){
65         vetor[i] = rand() % 100;
66     }
67     return vetor;
68 }
69
70 void desaloca(int *v){
71     free(v);
72 }
```

../q2_l1.c

Saída

```

zonzin@rodrigo:~/Documentos/Faculdade/labaedsii/labaeds2/pr1$ ./q2 12
Maior: 74
Menor: 1
Media: 28.08
zonzin@rodrigo:~/Documentos/Faculdade/labaedsii/labaeds2/pr1$ ./q2 15
Maior: 98
Menor: 1
Media: 53.93
zonzin@rodrigo:~/Documentos/Faculdade/labaedsii/labaeds2/pr1$ ./q2 3
Maior: 84
Menor: 7
Media: 38.00
zonzin@rodrigo:~/Documentos/Faculdade/labaedsii/labaeds2/pr1$ ./q2 4
Maior: 94
Menor: 0
Media: 34.25

```

Figura 2: Q2

1.3

Código

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  struct aluno{
6      char nome[20];
7      char matricula[10];
8      float nota;
9  };
10 typedef struct aluno Aluno;
11
12 void construtor(Aluno*, char*, char*,
13     float);
14 void imprime_aluno(Aluno*);
15 void pesquisa_max_min(Aluno**, int);
16
17 int main(){
18     Aluno **va =
19         (Aluno**)malloc(sizeof(Aluno)*3);
20     va[0] =
21         (Aluno*)malloc(sizeof(Aluno));
22     construtor(va[0], "Rodrigo",
23         "212050002", 1.0);
24     va[1] =
25         (Aluno*)malloc(sizeof(Aluno));
26     construtor(va[1], "Adelson",
27         "212050019", 0.5);
28     va[2] =
29         (Aluno*)malloc(sizeof(Aluno));
30     construtor(va[2], "Perozzo",
31         "te_amo", 05.072022);
32     pesquisa_max_min(va, 3);
33     free(va[2]);
34     free(va[1]);
35
36     free(va[0]);
37     free(va);
38     return 0;
39 }
40
41 void construtor(Aluno *a, char*nome,
42     char*matricula, float nota){
43     strcpy(a->nome, nome);
44     strcpy(a->matricula, matricula);
45     a->nota = nota;
46 }
47
48 void imprime_aluno(Aluno* a){
49     printf("Nome: %s\n", a->nome);
50     printf("Matricula: %s\n",
51         a->matricula);
52     printf("Nota: %.2f\n\n", a->nota);
53 }
54
55 void pesquisa_max_min(Aluno **va, int
56     n){
57     float maior = va[0]->nota,
58         menor = va[0]->nota;
59
60     int maxi = 0,
61         mini = 0;
62
63     for(int i = 0; i<n; i++){
64         if(va[i]->nota >= maior){
65             maior = va[i]->nota;
66             maxi = i;
67         }
68
69         if(va[i]->nota <= menor){
70             menor = va[i]->nota;
71             mini = i;
72         }
73     }
74 }

```

```

66                                     70     imprime_aluno(va[mini]);
67     printf("Maior: \n");           71 }
68     imprime_aluno(va[maxi]);
69     printf("Menor: \n");

```

../q3.l1.c

Saída

```

zonzin@rodrigo:~/Documentos/Faculdade/labaedst/labaeds2/pr1$ ./q3
Maior:
Nome: Perozzo
Matricula: te_amo
Nota: 5.07

Menor:
Nome: Adelson
Matricula: 212050019
Nota: 0.50

zonzin@rodrigo:~/Documentos/Faculdade/labaedst/labaeds2/pr1$ 

```

Figura 3: Q3

1.4

Código

ATENÇÃO: passe os valores a , b , e c como parâmetro da main.

Por exemplo, para calcularmos $x^2 + x - 12 = 0$, usamos:

`./q4 1 1 -12`

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <math.h>
5
6
7 double _delta(int b, int a, int c){
8     return (pow(b,2) - 4*a*c);
9 }
10
11 void bhaskara(int a, int b, int c,
12     double *x1, double *x2){
13     int delta = _delta(b,a,c);
14     if(delta<0) exit(1);
15     if(delta == 0.0){
16         *x1 = (double)(-b + pow(delta,
17             0.5) ) / (2*a);
18         *x1 = *x2;
19     }
20     if(delta > 0){
21         *x1 = (double)(-b + pow(delta,
22             0.5) ) / (2*a);
23         *x2 = (double)(-b - pow(delta,
24             0.5) ) / (2*a);
25     }
26 int main(int argc, char **argv){
27     double *x1 =
28         (double*)malloc(sizeof(double));
29     double *x2 =
30         (double*)malloc(sizeof(double));
31
32     int a = atoi(argv[1]),
33         b = atoi(argv[2]),
34         c = atoi(argv[3]);
35     bhaskara(a, b, c, x1, x2);
36     printf("x1 = %.21f\nx2 = %.21f\n",
37         *x1, *x2);
38
39     return 0;
40 }
```

../q4_l1.c

Saída

```
zonzin@rodrigo:~/Documentos/Faculdade/labaedsii/labaeds2/pr1$ ./q4 1 2 1
x1 = 0.00
x2 = 0.00
zonzin@rodrigo:~/Documentos/Faculdade/labaedsii/labaeds2/pr1$ ./q4 1 1 -6
x1 = 2.00
x2 = -3.00
zonzin@rodrigo:~/Documentos/Faculdade/labaedsii/labaeds2/pr1$ ./q4 1 6 14
zonzin@rodrigo:~/Documentos/Faculdade/labaedsii/labaeds2/pr1$ ./q4 -1 6 14
x1 = -1.80
x2 = 7.80
zonzin@rodrigo:~/Documentos/Faculdade/labaedsii/labaeds2/pr1$
```

Figura 4: Q4

2 Recursividade

2.1

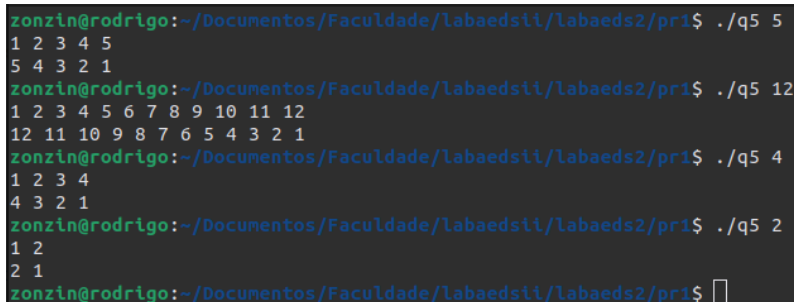
Código

ATENÇÃO: passar $n = 5$ como parâmetro no terminal.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4
5 void print_crescente(int n){
6     if(n == 0) return;
7     else print_crescente(n-1);
8     printf("%d ", n);
9 }
10
11 void print_decrescente(int n){
12     printf("%d ", n);
13     if(n == 1) return;
14     else print_decrescente(n-1);
15 }
16
17 int main(int argc, char const *argv[]){
18
19     int n = 5; //atoi(argv[1]);
20
21     print_crescente(n);
22     printf("\n");
23
24     print_decrescente(n);
25     printf("\n");
26
27     return 0;
28 }
```

../q5_l1.c

Saída

A terminal window with a dark background and green text. The prompt is 'zonzin@rodrigo:~/Documentos/Faculdade/labaedsti/labaeds2/pr1\$'. The user enters './q5 5', and the output is '1 2 3 4 5' followed by '5 4 3 2 1' on the next line. Then the user enters './q5 12', and the output is '1 2 3 4 5 6 7 8 9 10 11 12' followed by '12 11 10 9 8 7 6 5 4 3 2 1' on the next line. Then the user enters './q5 4', and the output is '1 2 3 4' followed by '4 3 2 1' on the next line. Then the user enters './q5 2', and the output is '1 2' followed by '2 1' on the next line. The prompt is visible at the end of the last line.

```
zonzin@rodrigo:~/Documentos/Faculdade/labaedsti/labaeds2/pr1$ ./q5 5
1 2 3 4 5
5 4 3 2 1
zonzin@rodrigo:~/Documentos/Faculdade/labaedsti/labaeds2/pr1$ ./q5 12
1 2 3 4 5 6 7 8 9 10 11 12
12 11 10 9 8 7 6 5 4 3 2 1
zonzin@rodrigo:~/Documentos/Faculdade/labaedsti/labaeds2/pr1$ ./q5 4
1 2 3 4
4 3 2 1
zonzin@rodrigo:~/Documentos/Faculdade/labaedsti/labaeds2/pr1$ ./q5 2
1 2
2 1
zonzin@rodrigo:~/Documentos/Faculdade/labaedsti/labaeds2/pr1$
```

Figura 5: Para uma entrada de $n = 5$ no terminal

2.2

Código

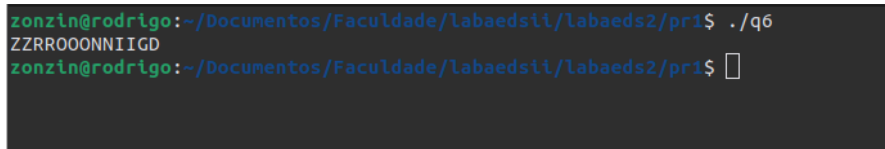
```

1 #include <stdlib.h>
2 #include <stdio.h>
3
4 void imprime_ordem_decrescente(char* vet, int pos_i){
5     if(pos_i < 0) return;
6     printf("%c", vet[pos_i]);
7     imprime_ordem_decrescente(vet, pos_i-1);
8 }
9
10 int comparaChar(const void* a, const void* b){
11     return (*(char*)a - *(char*)b);
12 }
13
14
15 int main(int argc, char** argv){
16     int n = 13;
17     char vet_palavras[13] = "RODRIGOZONZIN";
18
19     qsort(vet_palavras, n, sizeof(char), comparaChar);
20
21     imprime_ordem_decrescente(vet_palavras, n-1);
22     printf("\n");
23
24     return 0;
25 }

```

../q6_l1.c

Saída



```

zonzin@rodrigo:~/Documentos/Faculdade/labaeds11/labaeds2/pr1$ ./q6
ZZRROOONNIIGD
zonzin@rodrigo:~/Documentos/Faculdade/labaeds11/labaeds2/pr1$ █

```

Figura 6: Saída para a string "RODRIGOZONZIN"

2.3

ATENÇÃO: A implementação desse código é mais sutil. Seja a sequência natural $S = 3, 4, 5, 6, 7$. A recursividade deve partir do elemento S_n (o mais externo) até o elemento S_1 (o primeiro), da seguinte forma:

$$3 + 4 + 5 + 6 + 7$$

$$3 + 4 + 5 + 6$$

$$3 + 4 + 5$$

$$3 + 4$$

Para fazermos a soma de uma sequência de elementos naturais em um intervalo $[k, n]$ qualquer, nos valem da seguinte propriedade:

$$\sum_{i=k}^n = \sum_{i=1}^n - \sum_{i=1}^{k-1}$$

Dessa maneira, operamos a soma gaussiana sobre todos os subintervalos até que a diferença de um intervalo $[k_i, n_i] : n_i - k_1 \leq 0$, i. e., a condição de parada foi atingida.

Código

```
1 #include <stdlib.h>
2 #include <stdio.h>
3
4 int somatorio_gaussiano(int n){
5     return n*(n+1)/2;
6 }
7
8 int soma_intervalo(int i, int n){
9     return somatorio_gaussiano(n) - somatorio_gaussiano(i-1);
10 }
11
12 void imprime(int a, int b){
13     if(b-a <= 0) return;
14     printf("%d ", soma_intervalo(a, b));
15     imprime(a, b-1);
16 }
17
18 int main(int argc, char **argv){
19
20     int a = atoi(argv[1]),
21         b = atoi(argv[2]);
22
23     imprime(a, b);
24     printf("\n");
25
26     return 0;
27 }
```

../q7_l1.c

Saída


```

zonzin@rodrigo:~/Documentos/Faculdade/labaedsl/labaeds2/pr1$ ./q7 0 1
1
zonzin@rodrigo:~/Documentos/Faculdade/labaedsl/labaeds2/pr1$ ./q7 0 2
3 1
zonzin@rodrigo:~/Documentos/Faculdade/labaedsl/labaeds2/pr1$ ./q7 0 3
6 3 1
zonzin@rodrigo:~/Documentos/Faculdade/labaedsl/labaeds2/pr1$ ./q7 2 4
9 5
zonzin@rodrigo:~/Documentos/Faculdade/labaedsl/labaeds2/pr1$ ./q7 1 2
3
zonzin@rodrigo:~/Documentos/Faculdade/labaedsl/labaeds2/pr1$ 

```

Figura 7: Resultado para várias saídas

2.4

A implementação é trivial.

Código

```

1 #include <stdlib.h>
2 #include <stdio.h>
3
4 int mul(int x, int y){
5     if(y <= 0) return 0;
6
7     return x+mul(x, y-1);
8 }
9
10
11 int main(){
12     printf("%d \n", mul(10, 4));
13
14
15
16     return 0;
17 }

```

../q8_l1.c

Saída

2.5

Utilizamos a *flag* -555 para preencher o valor recursivamente.

Código

```

1 #include <stdlib.h>
2 #include <stdio.h>
3
4
5 void int_preenche(int* vet, int n){
6     if(n < 0) return;
7     vet[n] = -555;
8     int_preenche(vet, n-1);
9 }
10
11 void int_preenche_decresc(int* vet, int
    n){
12     if(n < 0) return;
13     vet[n] = n;
14     int_preenche_decresc(vet, n-1);
15 }
16
17 void int_preenche_cresc(int* vet, int
    n){
18     if(n < 0) return;
19     int_preenche_cresc(vet, n-1);
20     vet[n] = n;
21 }
22

```

```

23 int maior(int* vet, int n) {
24     if(n == 1) return vet[0];
25
26     int maior_corrente = maior(vet+1,
27                                 n-1);
28
29     return (vet[0] > maior_corrente) ?
30           vet[0] : maior_corrente;
31 }
32
33 int main(){
34     int n = 5;
35     int* vet =
36         (int*)malloc(sizeof(int)*n);
37
38     /* preenche */
39     int_preenche(vet,n-1);
40     for(int i =0; i<n; i++){
41         printf("%d ", vet[i]);
42     }
43     printf("\n");
44
45     /* preenche decrescente */
46     int_preenche_decresc(vet,n-1);
47     for(int i =0; i<n; i++){
48         printf("%d ", vet[i]);
49     }
50     printf("\n");
51
52     /*maior*/
53     printf("Maior: %d\n", maior(vet,
54                                 5));
55
56     free(vet);
57     return 0;
58 }
59
60
61 }

```

../q9.c

Saída

```

zonzin@rodrigo:~/Documentos/Faculdade/labaedsii/labaeds2/pr1$ ./q9
-555 -555 -555 -555 -555
0 1 2 3 4
0 1 2 3 4
Maior: 4
zonzin@rodrigo:~/Documentos/Faculdade/labaedsii/labaeds2/pr1$ 

```

Figura 8: Q9

2.6

O código foi analisado, executado e entendido para 1, 2, 3, 4 e 5 discos.