

## Guião 05 SO

Bernardo Mota Coelho

n.º mec: 125059

Turma P3

1.

a)

```
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05$ ./args1 111 222 333 1233 321
Argument 00: "./args1"
Argument 01: "111"
Argument 02: "222"
Argument 03: "333"
Argument 04: "1233"
Argument 05: "321"
```

O script imprime os argumentos em linhas diferentes e com o número da ordem dos mesmos.

b)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(int argc, char *argv[])
5  {
6      int i;
7
8      if (argc != 3) { // 1 Programa + 2 Argumentos = 3
9
10         printf("Insera só e apenas 2 argumentos \n");
11
12         return EXIT_FAILURE;
13     } else {
14
15         for(i = 0 ; i < argc ; i++)
16         {
17             printf("Argument %02d: \"%s\"\n", i, argv[i]);
18         }
19
20         return EXIT_SUCCESS;
21     }
22 }
```

bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05\$ gcc -Wall -o args1b args1b.c

Argument 00: "./args1b"

Argument 01: "123"

Argument 02: "123"

bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05\$ ./args1b 123

Insera só e apenas 2 argumentos

bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05\$ ./args1b

Insera só e apenas 2 argumentos

bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05\$ ./args1b 1 2 3

Insera só e apenas 2 argumentos

c)

```
aula05 > c calculadora.c > main(int, char * [])
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <math.h>
4
5  int main(int argc, char *argv[])
6  {
7
8      if (argc != 4) { // 1 Programa + 3 Argumentos = 4
9
10         printf("Insere 2 argumentos \n");
11
12         return EXIT_FAILURE;
13     } else {
14
15         //Calculadora
16         double result;
17         char op = argv[2][0];
18
19         double a = atof(argv[1]);
20         double b = atof(argv[3]);
21
22         switch (op)
23         {
24         case '+':
25             result = a + b;
26             break;
27         case '-':
28             result = a - b;
29             break;
30         case 'x':
31             result = a * b;
32             break;
33         case '/':
34             if (b == 0.0f) {
35                 printf("Erro: divisão por zero\n");
36                 return EXIT_FAILURE;
37             }
38             result = a / b;
39             break;
40         case 'p':
41             result = pow(a,b);
42             break;
43         default:
44             printf("Operador inválido: %c\n", op);
45             return EXIT_FAILURE;
46         }
47
48         printf("Resultado: %.2f \n", result);
49
50         return EXIT_SUCCESS;
51     }
52 }
```

d)

```

aula05 > C calculadora.c @ main(int, char *[])
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <math.h>
4  #include <errno.h>
5
6  int main(int argc, char *argv[])
7  {
8      if (argc != 4) {
9          printf("Uso: %s <número1> <operador> <número2>\n", argv[0]);
10         return EXIT_FAILURE;
11     }
12
13     double a, b, result;
14     char op = argv[2][0];
15     char *endptr;
16
17     errno = 0;
18     a = strtod(argv[1], &endptr);
19     if (errno == ERANGE || endptr == argv[1] || *endptr != '\0') {
20         printf("Erro: '%s' não é um número válido\n", argv[1]);
21         return EXIT_FAILURE;
22     }
23
24     errno = 0;
25     b = strtod(argv[3], &endptr);
26     if (errno == ERANGE || endptr == argv[3] || *endptr != '\0') {
27         printf("Erro: '%s' não é um número válido\n", argv[3]);
28         return EXIT_FAILURE;
29     }
30
31     switch (op) {
32         case '+':
33             result = a + b;
34             break;
35         case '-':
36             result = a - b;
37             break;
38         case 'x':
39             result = a * b;
40             break;
41         case '/':
42             if (b == 0.0) {
43                 printf("Erro: Divisão por zero\n");
44                 return EXIT_FAILURE;
45             }
46             result = a / b;
47             break;
48         case 'p':
49             result = pow(a, b);
50             break;
51         default:
52             printf("Erro: Operador '%c' inválido\n", op);
53             return EXIT_FAILURE;
54     }
55
56     printf("%.6g\n", result);
57     return EXIT_SUCCESS;
58 }

```

d)

O '\*' é interpretado pelo shell para listar ficheiros, por isso o programa recebe mais de 4 argumentos e mostra que dá erro.

2.

a)

```

bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula05$ ./args2
This program is being executed by bernardoc
All arguments have 0 characters
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula05$ ./args2 123
This program is being executed by bernardoc
All arguments have 3 characters
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula05$ ./args2 123 444 22 3312
This program is being executed by bernardoc
All arguments have 12 characters
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula05$ ./args2 123 4 5 6 78 9
This program is being executed by bernardoc
All arguments have 9 characters
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula05$

```

Imprime a soma de caracteres de todos os argumentos.

b)

```
5 int main(int argc, char **argv)
6 {
7     char *username;
8
9     username = getenv("NEWUSER");
10    if(username != NULL)
11    {
12        printf("This program is being executed by %s\n", username);
13    }
14    else
15    {
16        printf("ERROR: NEWUSER not defined\n");
17        return EXIT_FAILURE;
18    }
19
20    numChars = 0;
21    for(i = 1; i < argc; i++)
22    {
23        numChars += strlen(argv[i]);
24    }
25
26    printf("All arguments have %d characters\n", numChars);
27
28    return EXIT_SUCCESS;
29 }
30
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05$ NEWUSER = "AABBCC" ./args2b teste 123
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05$ NEWUSER="AABBCC" ./args2b teste 123
This program is being executed by AABBCC
All arguments have 8 characters
```

c)

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 int main(int argc, char *argv[])
6 {
7     if (argc < 2) {
8         printf("Uso: %s <palavras...>\n", argv[0]);
9         return EXIT_FAILURE;
10    }
11
12    int totalLength = 0;
13    for (int i = 1; i < argc; i++) {
14        totalLength += strlen(argv[i]) + 1; // +1 para o espaço ou '\0'
15    }
16
17    char *sentence = malloc(totalLength * sizeof(char));
18    if (sentence == NULL) {
19        perror("Erro a alocar memória");
20        return EXIT_FAILURE;
21    }
22
23    sentence[0] = '\0';
24    for (int i = 1; i < argc; i++) {
25        strcat(sentence, argv[i]);
26        if (i < argc - 1)
27            strcat(sentence, " ");
28    }
29
30    printf("%s\n", sentence);
31
32    free(sentence);
33    return EXIT_SUCCESS;
34 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05$ ./joinWords 123 123
123 123
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05$ ./joinWords 123 123 asd bdddr3ef
123 123 asd bdddr3ef
```

d)

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <ctype.h>
5 int main(int argc, char *argv[])
6 {
7     if (argc < 2) {
8         printf("Uso: %s <palavras...>\n", argv[0]);
9         return EXIT_FAILURE;
10    }
11
12    int totalLength = 0;
13
14    for (int i = 1; i < argc; i++) {
15        if (isalpha((unsigned char)argv[i][0])) {
16            totalLength += strlen(argv[i]) + 1;
17        }
18    }
19
20    if (totalLength == 0) {
21        printf("Nenhum argumento válido.\n");
22        return EXIT_SUCCESS;
23    }
24
25    char *sentence = malloc(totalLength * sizeof(char));
26    if (sentence == NULL) {
27        perror("Erro a alocar memória");
28        return EXIT_FAILURE;
29    }
30
31    sentence[0] = '\0';
32
33    for (int i = 1; i < argc; i++) {
34        if (isalpha((unsigned char)argv[i][0])) {
35            strcat(sentence, argv[i]);
36            strcat(sentence, " ");
37        }
38    }
39
40    if (strlen(sentence) > 0 && sentence[strlen(sentence) - 1] == ' ')
41        sentence[strlen(sentence) - 1] = '\0';
42
43    printf("%s\n", sentence);
44    free(sentence);
45    return EXIT_SUCCESS;
46 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

bernardoc@BernardoC-ZBook:~/UA/2ano/Semestre/50/mulad05$ gcc joinWordsText.c -o joinWordsText
abccddeejd nuncjuaedenj
bernardoc@BernardoC-ZBook:~/UA/2ano/Semestre/50/mulad05$ ./joinWordsText abccddeejd nuncjuaedenj 12312 123 123
abccddeejd nuncjuaedenj
bernardoc@BernardoC-ZBook:~/UA/2ano/Semestre/50/mulad05$ ./joinWordsText abccddeejdV4s 2 3 cs nuncjuaedenj 12312 123 123
abccddeejdV4s cs nuncjuaedenj
bernardoc@BernardoC-ZBook:~/UA/2ano/Semestre/50/mulad05$ ./joinWordsText abccddeejdV4s 2323bbccc 3 cs nuncjuaedenj 12312 123 123
abccddeejdV4s cs nuncjuaedenj
bernardoc@BernardoC-ZBook:~/UA/2ano/Semestre/50/mulad05$ ./joinWordsText 11bbb 11bb111
11bb111

```

3.

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4
5 int main(int argc, char *argv[])
6 {
7     if (argc != 3) {
8         printf("Uso: %s <limite_inferior> <limite_superior>\n", argv[0]);
9         return EXIT_FAILURE;
10    }
11
12    int min = atoi(argv[1]);
13    int max = atoi(argv[2]);
14
15    if (min >= max) {
16        printf("Erro: o limite inferior deve ser menor que o superior.\n");
17        return EXIT_FAILURE;
18    }
19
20    srand(time(NULL));
21
22    int secreto = rand() % (max - min + 1) + min;
23    int tentativa, tentativas = 0;
24
25    printf("Adivinhe o número entre %d e %d!\n", min, max);
26
27    do {
28        printf("Introduza o seu palpite: ");
29        if (scanf("%d", &tentativa) != 1) {
30            printf("Entrada inválida. Introduza um número inteiro.\n");
31            while (getchar() != '\n');
32            continue;
33        }
34
35        tentativas++;
36
37        if (tentativa < secreto)
38            printf("Mais alto!\n");
39        else if (tentativa > secreto)
40            printf("Mais baixo!\n");
41        else
42            printf("Acertou em %d tentativas!\n", tentativas);
43    } while (tentativa != secreto);
44 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

bernardoc@BernardoC-ZBook:~/UA/2ano/Semestre/50/aula05$ ./altobaixo 4 6
Adivinhe o número entre 4 e 6!
Introduza o seu palpite: 5
Mais baixo!
Introduza o seu palpite: 4
Acertou em 2 tentativas!
bernardoc@BernardoC-ZBook:~/UA/2ano/Semestre/50/aula05$ ./altobaixo 10 100
Adivinhe o número entre 10 e 100!
Introduza o seu palpite: 55
Mais alto!
Introduza o seu palpite: 70
Mais alto!
Introduza o seu palpite: 85
Mais alto!
Introduza o seu palpite: 90
Mais alto!
Introduza o seu palpite: 96
Mais baixo!
Introduza o seu palpite: 93
Mais baixo!
Introduza o seu palpite: 92
Acertou em 7 tentativas!

```

4.  
a)

```
C joinWordsText.c U C altoBaixo.c U C sortWords.c U x
aula05 > C sortWords.c > @ main(int, char *[])
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <ctype.h>
5
6 // Função de comparação (case-insensitive)
7 int compareAsc(const void *a, const void *b) {
8     const char *s1 = *(const char **)a;
9     const char *s2 = *(const char **)b;
10    return strcasecmp(s1, s2); // ignora maiúsculas/minúsculas
11 }
12
13 int compareDesc(const void *a, const void *b) {
14     const char *s1 = *(const char **)a;
15     const char *s2 = *(const char **)b;
16    return strcasecmp(s2, s1); // ordem inversa
17 }
18
19 int main(int argc, char *argv[])
20 {
21     if (argc < 2) {
22         printf("Uso: %s <palavras...>\n", argv[0]);
23         return EXIT_FAILURE;
24     }
25
26     // Filtrar apenas palavras que começam por letra
27     char *validWords[argc - 1];
28     int count = 0;
29
30     for (int i = 1; i < argc; i++) {
31         if (isalpha((unsigned char)argv[i][0])) {
32             validWords[count++] = argv[i];
33         }
34     }
35
36     if (count == 0) {
37         printf("Nenhuma palavra válida.\n");
38         return EXIT_SUCCESS;
39     }
40
41     // Ler variável de ambiente SORTORDER
42     char *order = getenv("SORTORDER");
43
44     // Ordenar as palavras (ASC por omissão)
45     if (order != NULL && strcmp(order, "DESC") == 0) {
46         qsort(validWords, count, sizeof(char *), compareDesc);
47     } else {
48         qsort(validWords, count, sizeof(char *), compareAsc);
49     }
50
51     // Imprimir resultado
52     printf("Palavras ordenadas (%s):\n", (order != NULL ? order : "ASC"));
53     for (int i = 0; i < count; i++) {
54         printf("%s\n", validWords[i]);
55     }
56
57     return EXIT_SUCCESS;
58 }
59
60 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
61
62 * bernardoc@BernardoC-ZBook:~/IA/Zano/1semestre/S0/aula05$ gcc sortWords.c -o sortWords
63 * bernardoc@BernardoC-ZBook:~/IA/Zano/1semestre/S0/aula05$ unset SORTORDER
64 * bernardoc@BernardoC-ZBook:~/IA/Zano/1semestre/S0/aula05$ ./sortWords abb bbc ccc zzz dd
65 Palavras ordenadas (ASC):
66 abb
67 bbc
68 ccc
69 dd
70 zzz
71 * bernardoc@BernardoC-ZBook:~/IA/Zano/1semestre/S0/aula05$ unset SORTORDER=DESC
72 * bernardoc@BernardoC-ZBook:~/IA/Zano/1semestre/S0/aula05$ ./sortWords abb bbc ccc zzz dd
73 Palavras ordenadas (ASC):
74 abb
75 bbc
76 ccc
77 dd
78 zzz
79 * bernardoc@BernardoC-ZBook:~/IA/Zano/1semestre/S0/aula05$ export SORTORDER=DESC
80 * bernardoc@BernardoC-ZBook:~/IA/Zano/1semestre/S0/aula05$ ./sortWords abb bbc ccc zzz dd
81 Palavras ordenadas (DESC):
82 zzz
83 dd
84 ccc
85 bbc
86 abb
87 * bernardoc@BernardoC-ZBook:~/IA/Zano/1semestre/S0/aula05$
```

b)

```
aula05 > C sortWords2.c > @ main(void)
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <ctype.h>
5
6 // Função de comparação (ordem ascendente - ignora maiúsculas/minúsculas)
7 int compareAsc(const void *a, const void *b) {
8     const char *s1 = *(const char **)a;
9     const char *s2 = *(const char **)b;
10    return strcasecmp(s1, s2);
11 }
12
13 // Função de comparação (ordem descendente)
14 int compareDesc(const void *a, const void *b) {
15     const char *s1 = *(const char **)a;
16     const char *s2 = *(const char **)b;
17    return strcasecmp(s2, s1);
18 }
19
20 int main(void) {
```

```

aula05 > C sortWords2.c > main(void)
20 int main(void) {
21     while (1) {
22         printf("> ");
23         if (fgets(input, sizeof(input), stdin) == NULL)
24             break; // EOF
25
26         // Remover '\n' do final
27         input[strcspn(input, "\n")] = '\0';
28
29         // Linha vazia -> parar
30         if (strlen(input) == 0)
31             break;
32
33         // Só aceitar se começar por letra
34         if (isalpha((unsigned char)input[0])) {
35             // Guardar cópia da palavra
36             words[count] = strdup(input);
37             if (words[count] == NULL) {
38                 perror("Erro de memória");
39                 return EXIT_FAILURE;
40             }
41             count++;
42         }
43     }
44
45     if (count == 0) {
46         printf("Nenhuma palavra válida.\n");
47         return EXIT_SUCCESS;
48     }
49
50     // Ler variável de ambiente SORTORDER
51     char *order = getenv("SORTORDER");
52
53     // Ordenar as palavras (ASC por omissão)
54     if (order != NULL && strcasecmp(order, "DESC") == 0) {
55         qsort(words, count, sizeof(char *), compareDesc);
56     } else {
57         qsort(words, count, sizeof(char *), compareAsc);
58     }
59
60     // Mostrar resultado
61     printf("\nPalavras ordenadas (%s):\n", (order != NULL ? order : "ASC"));
62     for (int i = 0; i < count; i++) {
63         printf("%s\n", words[i]);
64         free(words[i]); // libertar memória alocada
65     }
66
67     return EXIT_SUCCESS;
68 }
69
70
71

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/S0/aula05$ export SORTORDER=ASC
./sortWords2

```

```

Palavra 3: banana
Palavra 4: uva
Palavra 5: maça
Palavra 6:
Palavra 6: fim

```

```

=== Palavras lidas: 5 ===
Ordenação: ASCENDENTE (SORTORDER=ASC)

```

```

=== Palavras ordenadas ===
1. ananas
2. banana
3. kiwi
4. maça
5. uva

```