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)

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

int main(int argc, char *argv[])
{
    int i;

    if (argc != 3) { // 1 Progrma + 2 Argumentos = 3

        printf("Insere só e apenas 2 argumentos \n");

        return EXIT_FAILURE;
    } else {

    for(i = 0 ; i < argc ; i++)
    {
        printf("Argument %02d: \"%s\"\n", i, argv[i]);
    }

    return EXIT_SUCCESS;
    }
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

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
Insere só e apenas 2 argumentos
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05$ ./args1b
Insere só e apenas 2 argumentos
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05$ ./args1b 1 2 3
Insere só e apenas 2 argumentos
```

c)

aula05 > C calculadora.c > main(int, char * [])

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

int main(int argc, char *argv[])
{

    if (argc != 4) { // 1 Progrma + 3 Argumentos = 4

        printf("Insere 2 argumentos \n");

        return EXIT_FAILURE;
    } else {

    //Calculadora
    double result;
    char op = argv[2][0];

    double a = atof(argv[1]);
    double b = atof(argv[3]);

    switch (op)
    {
    case '+':
        result = a + b;
        break;
    case '-':
        result = a - b;
        break;
    case 'x':
        result = a * b;
        break;
    case '/':
        if (b == 0.0f) {
            printf("Erro: divisão por zero\n");
            return EXIT_FAILURE;
        }
        result = a / b;
        break;
    case 'p':
        result = pow(a,b);
        break;
    default:
        printf("Operador inválido: %c\n", op);
        return EXIT_FAILURE;
    }

    printf("Resultado: %.2f \n", result);

    return EXIT_SUCCESS;
    }
}
```

d)

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

int main(int argc, char *argv[])
{
    if (argc != 4) {
        printf("Uso: %s <número1> <operador> <número2>\n", argv[0]);
        return EXIT_FAILURE;
    }

    double a, b, result;
    char op = argv[2][0];
    char *endptr;

    errno = 0;
    a = strtod(argv[1], &endptr);
    if (errno == ERANGE || endptr == argv[1] || *endptr != '\0') {
        printf("Erro: '%s' não é um número válido\n", argv[1]);
        return EXIT_FAILURE;
    }

    errno = 0;
    b = strtod(argv[3], &endptr);
    if (errno == ERANGE || endptr == argv[3] || *endptr != '\0') {
        printf("Erro: '%s' não é um número válido\n", argv[3]);
        return EXIT_FAILURE;
    }

    switch (op) {
        case '+':
            result = a + b;
            break;
        case '-':
            result = a - b;
            break;
        case 'x':
            result = a * b;
            break;
        case '/':
            if (b == 0.0) {
                printf("Erro: Divisão por zero\n");
                return EXIT_FAILURE;
            }
            result = a / b;
            break;
        case 'p':
            result = pow(a, b);
            break;
        default:
            printf("Erro: Operador '%c' inválido\n", op);
            return EXIT_FAILURE;
    }

    printf("%.6g\n", result);
    return EXIT_SUCCESS;
}
```

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

Imprime a soma de caracteres de todos os argumentos.

b)

```c
int main(int argc, char **argv)
{
    char *username;

    username = getenv("NEWUSER");
    if(username != NULL)
    {
        printf("This program is being executed by %s\n", username);
    }
    else
    {
        printf("ERROR: NEWUSER not defined\n");
        return EXIT_FAILURE;
    }

    numChars = 0;
    for(i = 1 ; i < argc ; i++)
    {
        numChars += strlen(argv[i]);
    }

    printf("All arguments have %d characters\n", numChars);

    return EXIT_SUCCESS;
}
```

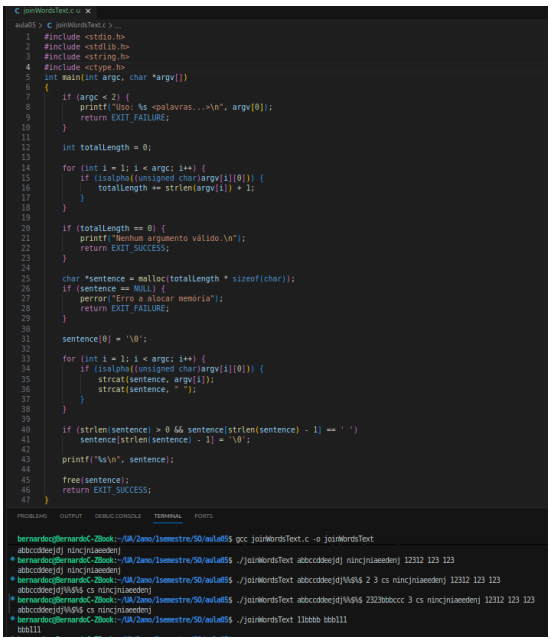
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05$ NEWUSER = "AABBCC" ./args2b teste 123
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05$ NEWUSER="AABBCC" ./args2b teste 123
This program is being executed by AABBCC
All arguments have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

c)

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

int main(int argc, char *argv[])
{
    if (argc < 2) {
        printf("Uso: %s <palavras...>\n", argv[0]);
        return EXIT_FAILURE;
    }

    int totalLength = 0;
    for (int i = 1; i < argc; i++) {
        totalLength += strlen(argv[i]) + 1; // +1 para o espaço ou '\0'
    }

    char *sentence = malloc(totalLength * sizeof(char));
    if (sentence == NULL) {
        perror("Erro a alocar memória");
        return EXIT_FAILURE;
    }

    sentence[0] = '\0';
    for (int i = 1; i < argc; i++) {
        strcat(sentence, argv[i]);
        if (i < argc - 1)
            strcat(sentence, " ");
    }

    printf("%s\n", sentence);

    free(sentence);
    return EXIT_SUCCESS;
}
```

```
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 bbddr3ef
123 123 asd bbddr3ef
```

d)



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
int main(int argc, char *argv[])
{
    if (argc < 2) {
        printf("Uso: %s <palavras...>\n", argv[0]);
        return EXIT_FAILURE;
    }

    int totalLength = 0;

    for (int i = 1; i < argc; i++) {
        if (isalpha((unsigned char)argv[i][0])) {
            totalLength += strlen(argv[i]) + 1;
        }
    }

    if (totalLength == 0) {
        printf("Nenhum argumento válido.\n");
        return EXIT_SUCCESS;
    }

    char *sentence = malloc(totalLength * sizeof(char));
    if (sentence == NULL) {
        perror("Erro a alocar memória");
        return EXIT_FAILURE;
    }

    sentence[0] = '\0';

    for (int i = 1; i < argc; i++) {
        if (isalpha((unsigned char)argv[i][0])) {
            strcat(sentence, argv[i]);
            strcat(sentence, " ");
        }
    }

    if (strlen(sentence) > 0 && sentence[strlen(sentence) - 1] == ' ')
        sentence[strlen(sentence) - 1] = '\0';

    printf("%s\n", sentence);

    free(sentence);
    return EXIT_SUCCESS;
}
```

```
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05$ gcc joinWordsText.c -o joinWordsText
abbccddeejdj nincjniaeedenj
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05$ ./joinWordsText abbccddeejdj nincjniaeedenj 12312 123 123
abbccddeejdj nincjniaeedenj
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05$ ./joinWordsText abbccddeejdj%&%$ 2 3 cs nincjniaeedenj 12312 123 123
abbccddeejdj%&%$ cs nincjniaeedenj
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05$ ./joinWordsText abbccddeejdj%&%$ 2323bbbccc 3 cs nincjniaeedenj 12312 123 123
abbccddeejdj%&%$ cs nincjniaeedenj
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05$ ./joinWordsText 11bbbb bbb111
bbb111
```

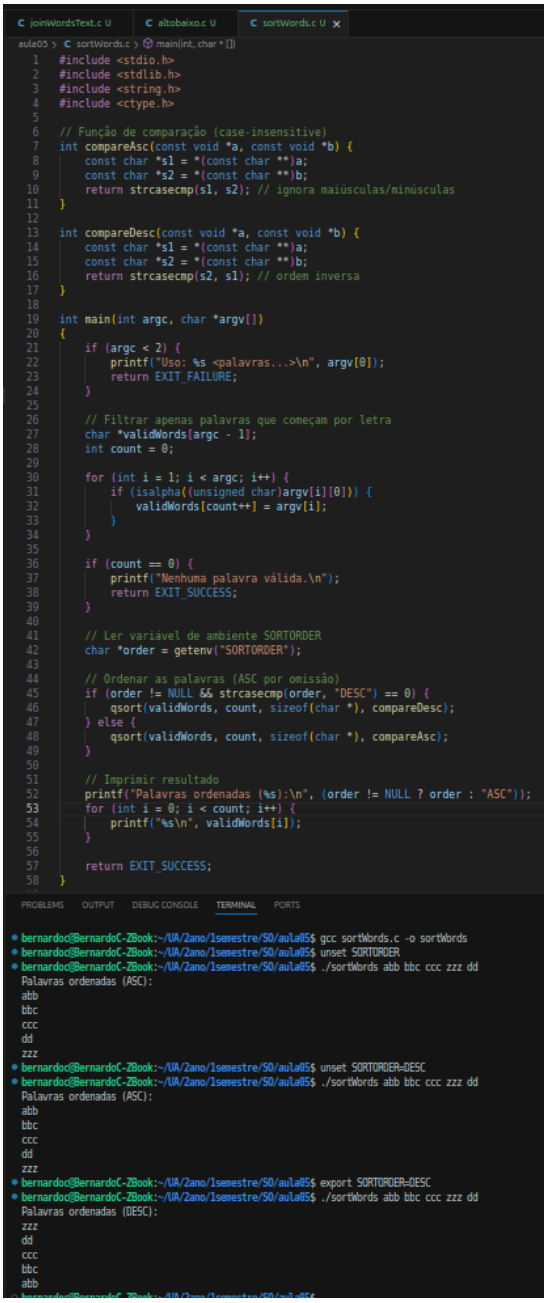
3.

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

int main(int argc, char *argv[])
{
    if (argc != 3) {
        printf("Uso: %s <limite_inferior> <limite_superior>\n", argv[0]);
        return EXIT_FAILURE;
    }

    int min = atoi(argv[1]);
    int max = atoi(argv[2]);

    if (min >= max) {
        printf("Erro: o limite inferior deve ser menor que o superior.\n");
        return EXIT_FAILURE;
    }

    srand(time(NULL));

    int secreto = rand() % (max - min + 1) + min;
    int tentativa, tentativas = 0;

    printf("Adivinhe o número entre %d e %d!\n", min, max);

    do {
        printf("Introduza o seu palpite: ");
        if (scanf("%d", &tentativa) != 1) {
            printf("Entrada inválida. Introduza um número inteiro.\n");
            while (getchar() != '\n');
            continue;
        }

        tentativas++;

        if (tentativa < secreto)
            printf("Mais alto!\n");
        else if (tentativa > secreto)
            printf("Mais baixo!\n");
        else
            printf("Acertou em %d tentativas!\n", tentativas);

    } while (tentativa != secreto);
```

```
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/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/1semestre/SO/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
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

// Função de comparação (case-insensitive)
int compareAsc(const void *a, const void *b) {
    const char *s1 = *(const char **)a;
    const char *s2 = *(const char **)b;
    return strcasecmp(s1, s2); // ignora maiúsculas/minúsculas
}

int compareDesc(const void *a, const void *b) {
    const char *s1 = *(const char **)a;
    const char *s2 = *(const char **)b;
    return strcasecmp(s2, s1); // ordem inversa
}

int main(int argc, char *argv[])
{
    if (argc < 2) {
        printf("Uso: %s <palavras...>\n", argv[0]);
        return EXIT_FAILURE;
    }

    // Filtrar apenas palavras que começam por letra
    char *validWords[argc - 1];
    int count = 0;

    for (int i = 1; i < argc; i++) {
        if (isalpha((unsigned char)argv[i][0])) {
            validWords[count++] = argv[i];
        }
    }

    if (count == 0) {
        printf("Nenhuma palavra válida.\n");
        return EXIT_SUCCESS;
    }

    // Ler variável de ambiente SORTORDER
    char *order = getenv("SORTORDER");

    // Ordenar as palavras (ASC por omissão)
    if (order != NULL && strcasecmp(order, "DESC") == 0) {
        qsort(validWords, count, sizeof(char *), compareDesc);
    } else {
        qsort(validWords, count, sizeof(char *), compareAsc);
    }

    // Imprimir resultado
    printf("Palavras ordenadas (%s):\n", (order != NULL ? order : "ASC"));
    for (int i = 0; i < count; i++) {
        printf("%s\n", validWords[i]);
    }

    return EXIT_SUCCESS;
}
```

```
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05$ gcc sortWords.c -o sortWords
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05$ unset SORTORDER
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05$ ./sortWords abb bbc ccc zzz dd
Palavras ordenadas (ASC):
abb
bbc
ccc
dd
zzz
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05$ unset SORTORDER=DESC
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05$ ./sortWords abb bbc ccc zzz dd
Palavras ordenadas (ASC):
abb
bbc
ccc
dd
zzz
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05$ export SORTORDER=DESC
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/aula05$ ./sortWords abb bbc ccc zzz dd
Palavras ordenadas (DESC):
zzz
dd
ccc
bbc
abb
```

b)

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

// Função de comparação (ordem ascendente - ignora maiúsculas/minúsculas)
int compareAsc(const void *a, const void *b) {
    const char *s1 = *(const char **)a;
    const char *s2 = *(const char **)b;
    return strcasecmp(s1, s2);
}

// Função de comparação (ordem descendente)
int compareDesc(const void *a, const void *b) {
    const char *s1 = *(const char **)a;
    const char *s2 = *(const char **)b;
    return strcasecmp(s2, s1);
}

int main(void) {
```

aula05 > C sortWords2.c > main(void)

```c
int main(void) {
    while (1) {
        printf("> ");
        if (fgets(input, sizeof(input), stdin) == NULL)
            break; // EOF

        // Remover '\n' do final
        input[strcspn(input, "\n")] = '\0';

        // Linha vazia -> parar
        if (strlen(input) == 0)
            break;

        // Só aceitar se começar por letra
        if (isalpha((unsigned char)input[0])) {
            // Guardar cópia da palavra
            words[count] = strdup(input);
            if (words[count] == NULL) {
                perror("Erro de memória");
                return EXIT_FAILURE;
            }
            count++;
        }
    }

    if (count == 0) {
        printf("Nenhuma palavra válida.\n");
        return EXIT_SUCCESS;
    }

    // Ler variável de ambiente SORTORDER
    char *order = getenv("SORTORDER");

    // Ordenar as palavras (ASC por omissão)
    if (order != NULL && strcasecmp(order, "DESC") == 0) {
        qsort(words, count, sizeof(char *), compareDesc);
    } else {
        qsort(words, count, sizeof(char *), compareAsc);
    }

    // Mostrar resultado
    printf("\nPalavras ordenadas (%s):\n", (order != NULL ? order : "ASC"));
    for (int i = 0; i < count; i++) {
        printf("%s\n", words[i]);
        free(words[i]); // libertar memória alocada
    }

    return EXIT_SUCCESS;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
bernardoc@BernardoC-ZBook:~/UA/2ano/1semestre/SO/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
```