Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Escola Politécnica – Curso de Bacharelado em Engenharia de *Software*
Disciplina: Programação de *Software* Básico – Professor: Roland Teodorowitsch
18 de setembro de 2025

Lista de Exercícios 5 (2025/2)

Este trabalho consiste em resolver a lista de exercícios das páginas a seguir, em C.
Para a entrega devem ser seguidas as seguintes regras:

- criar um arquivo compactado no formato ZIP com o nome do aluno no formato *camelHump* (por exemplo, para João Pedro da Silva, usar `JoaoPedroDaSilva.zip`), **SEM SUBDIRETÓRIOS** e **APENAS COM OS ARQUIVOS DE CÓDIGO-FONTE** (NÃO envie quaisquer outros arquivos, como, por exemplo, arquivos .class, .ctxt, README.txt, .o ou executáveis);
- o código-fonte deve ser **ADEQUADAMENTE INDENTADO**;
- o arquivo compactado deve conter programas em C para resolver cada um dos exercícios, salvando o código-fonte em um arquivo com o nome `Exercicio` seguido do número do exercício com **TRÊS dígitos** (por exemplo, `Exercicio001.c`, `Exercicio002.c`, ..., `Exercicio050.c`, ..., `Exercicio101.c`, ...);
- **ATENÇÃO:** os exercícios NÃO seguem necessariamente uma sequência contínua, então tome cuidado de **USAR O NÚMERO CORRETO DO EXERCÍCIO NO RESPECTIVO ARQUIVO DE CÓDIGO-FONTE**;
- **NÃO USAR ACENTOS NO NOME DE ARQUIVOS E DE FUNÇÕES** ;
- no início de cada arquivo em C, incluir um comentário informando o nome do arquivo, o nome do autor, a finalidade do programa e a versão (ou data) de criação (ou atualização);
- quando houver dados a serem lidos, **LER OS DADOS SEMPRE NA MESMA ORDEM EM QUE ELES SÃO CITADOS NO ENUNCIADO**, escolhendo os tipos numéricos adequadamente (se houver dúvida entre usar um tipo inteiro ou ou tipo real, use os exemplos de entradas e saídas que aparecem após cada exercício);
- **ESCREVER OS RESULTADOS SEMPRE NA MESMA ORDEM EM QUE ELES SÃO CITADOS NO ENUNCIADO**, escolhendo os tipos numéricos adequadamente (**NÚMEROS REAIS DEVEM SER IMPRESSOS SEMPRE COM 4 CASAS DECIMAIS**, a não ser que seja explicitamente indicado de outra forma);
- na versão final, tomar o cuidado de **NÃO IMPRIMIR NADA DIFERENTE DA SAÍDA ESPERADA** (não devem aparecer, por exemplo, mensagens pedindo que o usuário forneça ou digite determinado valor no terminal);
- a entrega deverá ser feita no dia e horário informado pelo professor em sala de aula e/ou definida na opção de entrega da plataforma moodle da PUCRS.

3. Alguns microcomputadores pessoais de 8 *bits*, bastante populares durante a década de 1980, trabalhavam com telas e imagens que possuíam um modo de 256 cores. Neste modo, usava-se uma paleta fixa em que cada cor era representada por um número de 8 bits, composto por 3 *bits* para verde, 3 *bits* para vermelho e 2 *bits* para azul. Para compor os campos de cada componente de verde, vermelho e azul, consideram-se apenas os *bits* mais significativos. Assim: da componente verde deve-se tomar os 3 *bits* mais significativos; da componente vermelha deve-se igualmente tomar os 3 *bits* mais significativos; e da componente azul deve-se tomar os 2 *bits* mais significativos.

   Uma cor com intensidade verde igual a 234 (em binário 11101010), intensidade vermelha igual a 74 (em binário 01001010) e intensidade azul igual a 213 (em binário 11010101) corresponde, então, ao valor decimal 235 (em binário 11101011).

   Escreva um programa que leia as intensidades de verde, vermelho e azul (todos como valores de 0 até 255) e gere o valor correspondente na paleta fixa de 256 cores para esse tipo de sistema.

   Exemplo(s):

Teste	Entrada	Saída
1	234 74 213	235
2	223 63 140	198
3	165 17 65	161
4	139 239 31	156
5	110 199 100	121
6	95 170 62	84
7	45 129 139	50
8	28 127 88	13
9	232 255 199	255
10	31 31 63	0

4. Alguns microcomputadores pessoais de 8 *bits*, bastante populares durante a década de 1980, trabalhavam com telas e imagens que possuíam um modo de 256 cores. Neste modo, usava-se uma paleta fixa em que cada cor era representada por um número de 8 bits, composto por 3 *bits* para verde, 3 *bits* para vermelho e 2 *bits* para azul. Para compor os campos de cada componente de verde, vermelho e azul, consideram-se apenas os *bits* mais significativos. Assim: da componente verde deve-se tomar os 3 *bits* mais significativos; da componente vermelha deve-se igualmente tomar os 3 *bits* mais significativos; e da componente azul deve-se tomar os 2 *bits* mais significativos.

   Uma cor com intensidade verde igual a 234 (em binário 11101010), intensidade vermelha igual a 74 (em binário 01001010) e intensidade azul igual a 213 (em binário 11010101) corresponde, então, ao valor decimal 235 (em binário 11101011).

   Por outro lado, quando se tem o valor de cor igual a 235 (em binário 11101011), por exemplo, os valores das componentes serão: 224 (em binário 11100000) para verde, 64 (em binário 01000000) para vermelho e 192 (em binário 11000000) para azul.

   Escreva um programa que leia o valor correspondente na paleta fixa de 256 cores para esse tipo de sistema e mostre as suas intensidades de verde, vermelho e azul.

   Exemplo(s):

Teste	Entrada	Saída
1	235	224 64 192
2	198	192 32 128
3	161	160 0 64
4	156	128 224 0
5	123	96 192 192
6	86	64 160 128
7	49	32 128 64
8	12	0 96 0
9	255	224 224 192
10	0	0 0 0

5. Em alguns dos primeiros microcomputadores pessoais, que usavam processadores de 8 *bits*, alguns programas (principalmente jogos) redefiniam os caracteres de texto usando um padrão de 8 por 8 *pixels* para cada caractere. Assim, para cada caractere eram definidos 8 *bytes* que constituiam o padrão de pontos usado para desenhar esse caractere. Basicamente o valor de cada *byte* deve ser convertido para binário e, se o *bit* estivesse ligado, um ponto era pintado; se o *bit* estivesse desligado, o ponto não era pintado.

   Por exemplo, um caractere definido como 112 (em binário 01110000), 56 (em binário 00111000), 56 (em binário 00111000), 108 (em binário 01101100), 124 (em binário 01111100), 198 (em binário 11000110), 238 (em binário 11101110) e 0 (em binário 00000000), ...

   Exemplo(s):

Teste	Entrada	Saída
1	112 56 56 108 124 198 238 0	.XXX.... ..XXX... ..XXX... .XX.XX.. .XXXXX.. XX...XX. XXX.XXX. ....
2	252 102 100 126 102 102 252 0	XXXXXX.. .XX..XX. .XX..X.. .XXXXXX. .XX..XX. .XX..XX. XXXXXX.. ....
3	124 226 192 192 192 226 124 0	.XXXXX.. XXX...X. XX...... XX...... XX...... XXX...X. .XXXXX.. ....
4	252 102 98 98 98 102 252 0	XXXXXX.. .XX..XX. .XX...X. .XX...X. .XX...X. .XX..XX. XXXXXX.. ....
5	252 100 96 120 96 98 254 0	XXXXXX.. .XX..X.. .XX..... .XXXX... .XX..... .XX...X. XXXXXXX. ....
6	252 100 96 120 96 96 240 0	XXXXXX.. .XX..X.. .XX..... .XXXX... .XX..... .XX..... XXXX.... ....
7	124 226 192 192 206 230 126 0	.XXXXX.. XXX...X. XX...... XX...... XX..XXX. XXX..XX. .XXXXXX. ....
8	238 108 108 124 108 108 238 0	XXX.XXX. .XX.XX.. .XX.XX.. .XXXXX.. .XX.XX.. .XX.XX.. XXX.XXX. ....
9	120 48 48 48 48 48 120 0	.XXXX... ..XX.... ..XX.... ..XX.... ..XX.... ..XX.... .XXXX... ....
10	120 48 48 48 48 48 48 96 0	.XXXX... ..XX.... ..XX.... ..XX.... ..XX.... ..XX.... ..XX.... .XX.....

6. Escreva um programa em C que leia um número inteiro e, usando operações *bit*-a-*bit*, conte e mostre quantos *bits* estão desligados na representação binária desse número, desconsiderando os bits à esquerda do *bit* 1 mais significativo (ou seja, seu programa deve contar e mostrar quantos *bits* 0 há à direita do *bit* 1 mais significativo).

   Assim, por exemplo, 20 em decimal corresponde a 10100 em binário, sendo que o número de *bits* 0 (desligados) à direita do *bit* 1 mais significativo é 3.

   Exemplo(s):

Teste	Entrada	Saída
1	20	3
2	12345	8
3	3333	7
4	-33	1
5	1	0
6	1024	10
7	54321	9
8	6789	7
9	0	0
10	2147483647	0
11	-2147483648	31
12	223344	10
13	-1	0

7. Escreva um programa em C que leia dois números inteiros decimais sem sinal. O primeiro número corresponde a um valor inteiro que ocupa no máximo 32 *bits* na memória. E o segundo número será sempre um valor de 0 até 31, que corresponde ao número do *bit* cujo valor se deseja obter e mostrar. Por exemplo, caso o primeiro número lido seja 2 (armazenado na memória em binário como `00000000 00000000 00000000 00000010`), e o segundo número seja `0`, o resultado deverá ser `0`, pois o *bit* menos significativo é `0`. Por outro lado, se primeiro número lido for `2` e o segundo número for `1`, o resultado deverá será `1`, pois o segundo *bit* menos significativo é `1`.

   Exemplo(s):

Teste	Entrada	Saída
1	2 0	0
2	2 1	1
3	12 0	0
4	12 1	0
5	12 2	1
6	12 3	1
7	12 4	0
8	2147483648 31	1
9	2147483648 30	0
10	1 31	0
11	1 0	1
12	1 1	0