EA871 – Laboratório de Programação Básica de Sistemas Digitais

Atividade 01

Profs. Levy Boccato, Rafael Ferrari e Tiago Tavares – 2º semestre de 2019

1. Objetivos

- Familiarização com a linguagem C e algoritmos essenciais.
- Utilizar variáveis, condicionais, laços e operações lógico-aritméticas.

2. Resumo da Atividade

Nesta atividade, aprenderemos a desenvolver, analisar e compilar códigos-fonte em linguagem C.

3. Roteiro da Aula

Exercício 1

Leia o código-fonte abaixo, correspondente ao programa 01-basico.c. Ele foi escrito em linguagem C.
a) Encontre todas as palavras-chave que você não conhece.
b) Encontre todas as linhas que são comentários.
c) Faça uma hipótese sobre o que o programa faz.
d) Compile e execute o programa. Ele faz o que você imaginava?

gcc 01-basico.c –o nome_arquivo_saida

Para executar, utilize o comando:

./nome_arquivo_saida

e) Modifique o programa de forma que ele imprima seu nome na tela.

```
#include <stdio.h>

int main() {
    /* printf("coisa a imprimir") imprime algo na tela.
        Alguns caracteres especiais podem ser uteis para formatacao:
        \n (newline) representa o fim de uma linha.
        \t (tabulation) representa uma tabulacao.
        \\ (barra invertida) imprime uma barra invertida.
    */

    printf("Estou funcionando!\n");
    return 0;
}
```

Exercício 2

Leia o código-fonte abaixo, correspondente a um trecho do programa 02-variaveis.c.
a) Encontre todas as palavras-chave que você não conhece.
b) Encontre todas as variáveis que foram declaradas no programa.
c) Encontre todas as operações de atribuição no programa.
d) Faça uma hipótese sobre o que o programa imprimirá na tela.
e) Compile e execute o programa. Ele imprimiu o que você imaginava?
f) Qual é o resultado de uma divisão com inteiros caso o resultado seja uma fração?
g) Por que é possível fazer aritmética com tipos char?

```
#include <stdio.h>

int main() {
    /* Neste programa, declararemos algumas variaveis:*/
```

```
    char c0, c1, c2, c3;  /* C permite a declaracao de varias variáveis numa unica sequencia */
    int i0, i1, i2, i3;
    float f0, f1, f2, f3;

    /* Vamos atribuir alguns valores e escreve-los na tela */
    c0 = 'd';
    i0 = 50;
    f0 = 50.0;

    /* Atencao a sintaxe do printf para escrever variaveis na tela!
        coloque um %c, %d ou %f para marcar, na string, as posicoes em que as
        variaveis serao escritas; apos, declare quais serao essas variaveis que
        serao escritas - na ordem que aparecem na string! */
    printf("Atribui os valores %c, %d e %f as minhas variaveis!\n", c0, i0, f0);

    /* Teste de aritmetica */
    printf("Aritmetica com int (+ - * / 3): %d, %d, %d, %d\n", i0+3, i0-3, i0*3, i0/3);
    printf("Aritmetica com float (+ - * / 3): %f, %f, %f, %f\n", f0+3, f0-3, f0*3, f0/3);
    printf("Aritmetica com char (+ - * / 3): %c, %c, %c, %c\n", c0+3, c0-3, c0*3, c0/3);

    /* Mas, lembre que char eh um tipo inteiro! Entao, o codigo seguinte eh valido: */
    printf("Aritmetica com char (+ - * / 3): %d, %d, %d, %d\n", c0+3, c0-3, c0*3, c0/3);
    return 0;
}
```

Exercício 3

Leia o código-fonte abaixo, correspondente a um trecho do programa 03-condicionais.c.

a) Encontre todas as palavras-chave que você não conhece.

b) Encontre todas as instruções condicionais no programa.

c) Faça uma hipótese sobre o que o programa imprimirá na tela.

d) Compile e execute o programa. Ele imprimiu o que você imaginava?

e) Use chaves { } para modificar o programa de forma que ele execute mais de uma instrução relacionada a cada avaliação de if(). As chaves { } em C são equivalentes à indentação em Python.

```
#include <stdio.h>

int main() {
    int a, b, c; /* Vamos utilizar estas variaveis para fazer testes */
    float x, y, z; /* Tambem vamos testar coisas com estas variaveis */

    printf("Comparacao - int com int\n");
    a = 0;
    if (0 == 0) printf("0 == 0\n");
    if (a == 0) printf("a == 0\n");
    if (a == 1) printf("a == 1\n");
    printf("\n");

    printf("Comparacao - int com float\n");
    x = 0.0;
    y = 0.5;
    if (0.0 == 0.0) printf("0.0 == 0.0\n");
    if (x == 0.0) printf("x == 0.0\n");
    if (a == x) printf("0 == 0.0\n");
    if (a == y) printf("0 == 0.5\n");
    printf("\n");

        return 0;
}
```

Exercício 4

Leia o código-fonte abaixo, correspondente a um trecho do programa 04-entradas_saidas.c.
a) Encontre todas as palavras-chave que você não conhece.
b) Um dos parâmetros da instrução scanf tem um símbolo &. O que ele significa?
c) Faça uma hipótese sobre o que o programa faz.
d) Compile e execute o programa. Ele fez que você imaginava?
e) Modifique o programa para que ele escreva "PASSOU" caso o número digitado pelo usuário seja maior ou igual a 5, e "REPROVOU" caso o número seja menor que 5.

```
#include <stdio.h>

int main() {
	int a;
	float b;

	printf("Digite um inteiro e entao ENTER\n");
	scanf("%d", &a);
	printf("%d\n", a);

	return 0;
}
```

Exercício 5

Leia o código-fonte abaixo, correspondente a um trecho do programa 05-lacos.c.
a) Encontre todas as palavras-chave que você não conhece.
b) Qual é a diferença entre os blocos do..while(), while() e for(;;)?
c) Faça uma hipótese sobre o que o programa faz.
d) Compile e execute o programa. Ele fez que você imaginava?

```
#include <stdio.h>

int main() {
	int i;
	int f;

	printf("Loops\n");

	printf("1) Contando de 1 a 15\n");
	i = 0;
	do {
		i = i + 1;
		printf("%d\t", i);
	} while (i<15);
	printf("\n");
	printf("\n");

	printf("2) Calculando fatorial de 5\n");
	i = 5;
	f = 1;
	while (i>0) {
		f = f * i;
		i = i - 1;
	}
	printf("O fatorial de 5 eh %d\n", f);
	printf("\n");
	printf("3) Imprimindo um tabuleiro de xadrez\n");
	printf("1 = casa branca, 0 = casa preta\n");
	for (i = 0; i < 8; i=i+1) {
		for (f = 0; f < 8; f=f+1) {
```

```
            printf("%d\t", ((i+f+1)%2)); /* Exercicio: explique porque (i+f+1)%2 */
        }
        printf("\n");
    }
    printf("\n");
    printf("4) Imprimindo uma piramide com 10 andares\n");
    for (i = 0; i < 10; i=i+1) { /* Para cada andar */
        for (f = 0; f <= i; f=f+1) { /* Exercicio: porque precisamos de f<=i e não f<i ? */
            printf("* ");
        }
        printf("\n");
    }
    printf("---\n");

    return 0;
}
```

Exercício 6

Analise o programa abaixo.
a) O que as operações **&**, | e **<<** fazem?
b) O que a notação **0x…** significa?
c) O que deverá ser impresso na tela?
d) Entre com o programa em seu editor preferido, compile e execute. Sua hipótese se confirmou?
e) Acrescente operações após a última atribuição de modo que o bit mais significativo da variável c (bit 7) seja 1, o bit 2 seja 0 e os demais permaneçam inalterados.

```
#include <stdio.h>

int main() {
        char c = 0xF3;
        c = c & 0x01;
        c = c << 3;
        c = c | 0x06;
        printf("%d\n", c);
        return 0;
}
```

4. Exercício computacional para casa (individual)

O objetivo desta tarefa é contar o número de bits iguais a 1 em um *byte* recebido como entrada. No programa fornecido como base, já está implementada a leitura de um valor recebido na entrada padrão (teclado) como hexadecimal e o armazenamento do resultado numa variável de 1 *byte* do tipo *unsigned char*. Além disso, a impressão do resultado já está implementada. Isso significa que a solução desta tarefa se concentra somente em implementar a contagem de bits.

Entrada (Hexadecimal)	Saída
00	0
FF	8
01	1
FE	7
80	1
7F	7
63	4

Instruções para a submissão do trabalho

1) Baixe o *template* da atividade 1 do Google Classroom.
2) Modifique o arquivo src/main.c para completar seu laboratório.
3) Use o comando **make** em uma janela do terminal aberta no diretório raiz para compilar seu código.
4) Use o comando **make test** para testar o funcionamento de seu programa.
5) Quando terminar, crie um arquivo no formato .zip (Aviso: não use .tar.gz nem .rar) cujo nome é seu_ra.zip (Exemplo: 025304.zip) com toda a estrutura de diretórios que você baixou.
6) Faça o *upload* da sua solução da atividade 1 no Google Classroom.