SCC0601 – Introdução à Ciência da Computação II
Prof. Lucas Antiqueira

1ª lista de exercícios

1) Em C, a quantidade de bytes para cada tipo de dado é dependente do sistema (varia). O operador `sizeof` pode ser usado para obter o tamanho, em bytes, dos tipos de dados. Verifique os tamanhos do tipos `char`, `int`, `float` e `double` no seu sistema.

```
#include <stdio.h>
int main (){
      printf("\n char: %d", sizeof(char)) ;
      printf("\n int: %d", sizeof(int)) ;
      printf("\n float: %d", sizeof(float)) ;
      printf("\n double: %d", sizeof(double)) ;
      return(0);
}
```

2) Considerando o seguinte programa, aponte e corrija um erro que está presente no código:

```
#include <stdio.h>
int main(void){
      int idade;
      char nome[20];
      printf("Digite o seu nome:\n");
      scanf("%s", &nome);
      printf("Digite sua idade: \n");
      scanf("%d", &idade);
      printf("\nOlá %s, seja bem-vindo!\n",nome);
      printf("Você nasceu em %d", 2011-idade);
      return(0);
}
```

3) Qual o resultado da variável `expr` após a execução do programa abaixo?

```
int main(void){
   int a, b, c, expr;
   a = 3;
   b = 4;
   c = 5;
   expr = (((b++)*(a-1))+((b*3) - (--c)))%(40/c);
   return expr;
}
```

4) Para cada expressão abaixo, diga se ela retornará verdadeiro (1) ou falso (0):

```
a) (10>5)&&(5>10)
b) (10>5)||(5>10)
c) ((25>26)||(27>27))&&((3!=2)||(3==3))
d) !((4<7)&&(6==5))||((14)&&(0))
e) (170&85)
f) (36|18)>>2
g) ((40^50)&(~50))<<2
```

5) Este exercício ilustra uma situação onde o espaço em branco ao redor de operadores é importante. A expressão `a+++b` pode ser interpretada como `a++ + b` ou `a + ++b`, dependendo de como o sinal de mais é agrupado. O agrupamento correto é o primeiro. O compilador agrupa a cadeia de caracteres mais longa possível para formar um símbolo, então usa ++ em vez + como o primeiro símbolo após a variável `a`. Escreva um programa para verificar esta propriedade.

6) Sendo `p` um ponteiro para um inteiro de 2 bytes, explique a diferença entre:

```
a) p++
b) (*p)++
c) *(p++)
d) *(p+10)
```

7) Se `i` e `j` são do tipo `int`, e `p` e `q` são ponteiros para `int`, quais das seguintes expressões não são válidas?

```
a) p = &j;
b) i = (*&)j;
c) p = &*&i;
d) i = *&*&j;
e) i = (int) p;
f) i = *p++ + *q;
g) q = &p *q = &j;
```

8) Verifique o programa abaixo. Encontre o seu erro e corrija-o para que escreva o numero 10 na tela (utilizando a variável `q`):

```
#include <stdio.h>
int main(void) {
    int x, *p, **q;
    p = &x;
    q = &p;
    x = 10;
    printf("\n%d\n", &q);
    return(0);
}
```

9) A operação `min(x,y)` pode ser representada pela expressão condicional `(x<y) ? x : y`. De modo semelhante, usando apenas expressões condicionais, represente as operações `min(x,y,z)` e `min(x,y,z,w)`.

10) Compile e execute o programa a seguir. Verifique quais valores foram impressos. Explique o porquê dos valores terem sido impressos da forma que foram, levando em conta as regras de escopo e a passagem de parâmetros.

```
#include <stdio.h>

int a=10;

void f1 (int a);
void f2 (int x);
void f3 (int x);

int main (void) {
    int a=20;
    printf("a em main (1): %d\n", a);
    f1(a);
    printf("a em main (2): %d\n", a);
    f2(a);
    f3(a);
    printf("a em main (3): %d\n", a);
    return 0;
}

void f1 (int a) {
    a += 10;
    printf("a em f1: %d\n", a);
}

void f2 (int x) {
    int a = x;
    printf("a em f2: %d\n", a);
}
```

```
void f3 (int x) {
    a = a + 100;
    {
        int a = 50;
        printf("a em f3 (2): %d\n", a);
        {
            a = a + 10;
            printf("a em f3 (3): %d\n", a);
        }
    }
    printf("a em f3 (4): %d e x: %d\n", a, x);
}
```

11) Escreva uma função que realize a concatenação de duas strings. A função deve receber, como parâmetros, as 2 strings a serem concatenadas e deve retornar uma nova string com o resultado.

12) Se `p` é um ponteiro então as duas expressões `*p++` e `(*p)++` tem diferentes resultados. Analise o código abaixo e diga o que será impresso. Explique.

```
char a[] = "abc";
char *p;
int i;

p = a;
for (i = 0; i < 3; ++i)
      printf("%c\n", *p++);
printf("a = %s\n", a);
p = a;
for (i = 0; i < 3; ++i)
      printf("%c\n", (*p)++);
printf("a = %s\n", a);
```

13) Crie uma estrutura chamada "ponto" que conterá os valores `x` e `y` de um ponto no plano cartesiano. Em seguida, crie estruturas derivadas da primeira para retângulos, círculos, quadrados e triângulos. Faça um programa que leia essas estruturas e retorne a respectiva área.

14) Seja uma matriz bidimensional de ordem `n` x `m` qualquer. Encontre uma forma de se armazenar os elementos dessa matriz em uma outra matriz unidimensional. Como deveria ser a fórmula para acessar um elemento que está na linha `i` e coluna `j` dessa matriz? E no caso de uma matriz `n`-dimensional?

15) Implemente uma matriz `M` x `N` (`M` e `N` são fornecidos pelo usuário) alocada dinamicamente.