Universidade do Estado de Santa Catarina – UDESC
Centro de Ciências Tecnológicas CCT

Prof. André Tavares da Silva
andre.silva@udesc.br

Exercícios de Linguagem C

1. O código abaixo contém a declaração de algumas variáveis e ponteiros. Quando você executa esse código, quais serão os valores de x, y e p ao final da execução do trecho de código abaixo?

```
void main() {
   int x, y, *p; y = 0;
   p = &y;
   x = *p;
   x = 4;
   (*p)++;
   --x;
   (*p) += x;
}
```

2. Uma das atividades mais desafiadoras é entender o código feito por outros programadores. Isso fica ainda pior quando existem erros ou "bugs" nos códigos. Identifique-os e reescreva indicando como deveriam ser?

a)

```
void main() {
   int x, *p;
   x = 100;
   p = x;
   printf("Valor de p: %d.n", *p);
}
```

b)

```
void troca (int *i, int *j) {
   int *temp;
   *temp = *i;
   *i = *j;
   *j = *temp;
}
```

c)

```
main(){
   char *a, *b;
   a = "abacate";
   b = "uva";
   if (a < b)
      printf ("%s vem antes de %s no dicionário", a, b);
   else
      printf ("%s vem depois de %s no dicionário", a, b);
}
```

3) Qual o conteúdo do vetor "a" depois dos seguintes comandos:

```
main() {
     int i, a[99];

     for (i = 0; i < 99; ++i)
          a[i] = 98 - i;

     for (i = 0; i < 99; ++i)
          a[i] = a[a[i]];
}
```

4) Analise o código abaixo e explique cada uma das linhas usando comentários.

```
main (){
   int x = 100, *p, **pp;
   p = &x;
   pp = &p;
   printf("Valor de pp: %dn", **pp);
}
```

5) O operador asterisco (*) serve para declaração de uma variável do tipo ponteiro e para fazer referência ao conteúdo de um ponteiro (acessar o valor). Qual significado do operador asterisco em cada um dos seguintes casos:

a) `int *p;`
b) `printf("%d",*p);`
c) `*p = x*5;`
d) `printf("%d",*(p+1));`

6) Os ponteiros são excelentes exercícios de lógica, a seguir temos uma função main com alguns ponteiros e variáveis. Identifique o que será impresso na tela.

```
void main(){
   int i=5, *p;
   p = &i;
   printf("%d, %d, %d ,%d, %d", p, (*p+2), **&p, (3**p), (**&p+4) );
}
```

7) Crie uma função que receba por parâmetro um vetor de números inteiros e os endereços de duas variáveis inteiras (que podemos chamar de menor e maior). Ao passar essas variáveis para a função seu programa deverá analisar qual é o maior e o menor elemento do vetor e depositar esses elementos nas variáveis do parâmetro. Crie uma função *main* que utilize a função que você definiu. Use o seguinte protótipo para sua função:

```
void maior_menor(int vetor[], int* menor, int* maior);
```

8) Crie uma função que copia um vetor de caracteres para outro vetor (uma cópia). A assinatura (protótipo) da função deve ser:

```
char *strcopy(char *str, int tamanho);
```

9) Vamos criar uma função agora que localiza uma letra em um vetor e retorna um outro vetor com suas posições onde a letra foi encontrada. Por exemplo:

```
  0   1   2   3   4   5   6   7   8   9  10  11
[ u | n | i | v | e | r | s | i | d | a | d | e ]

// resultado da busca pela letra "i"
[ 2 | 7 ]
```

10) Faça um programa que leia um valor n e crie dinamicamente um vetor de n elementos e passe esse vetor para uma função que vai ler os elementos desse vetor. Depois, no programa principal, o vetor preenchido deve ser impresso. Além disso, antes de finalizar o programa, deve-se liberar a área de memória alocada.

11) Criar um tipo abstrato de dados que represente uma pessoa, contendo nome, data de nascimento e CPF. Aloque dinamicamente uma variável desse novo tipo (na função principal). Depois crie uma função que receba este ponteiro e preencha os dados da estrutura. A seguir crie também uma uma função que receba este ponteiro e imprima os dados da estrutura. Finalmente, faça a chamada a esta função na função principal.

12) Implemente um algoritmos de busca sequencial em um vetor de inteiros. Exiba ao final quantos testes foram realizados para a busca. A função deve retornar 1 (um) se o valor estiver no vetor e 0 (zero) se não estiver presente.

```
int busca(int chave, int vetor[], int tamanho_vetor);
```

13) Faça uma função recursiva para realizar esta mesma busca.

14) Considerando o vetor ordenado, crie uma função que localiza um elemento e retorna em um outro vetor as suas posições onde o elemento foi encontrado.

```
int *busca(int chave, int vetor[], int tamanho_vetor);
```

15) Considere agora que o vetor está **ordenado**. Implemente a busca de um valor sabendo que o valor pode não estar no vetor. Exiba ao final quantos testes foram realizados para a busca. A função deve retornar 1 (um) se o valor estiver no vetor e 0 (zero) se não estiver presente.

```
int busca(int chave, int vetor[], int tamanho_vetor);
```

16) Implemente a busca binária utilizando recursividade para um vetor ordenado (ver resolução do exercício anterior).

17) Suponha que um vetor de inteiros contenha valores como: 1, 2, 4, 8, 16, 32, 64,(…). Crie um programa que implementa a busca binária para encontrar um valor no vetor (alternativamente, retorne o valor mais próximo do requerido).

18) Escreva um algoritmo recursivo para avaliar a * b usando a adição, onde a e b são inteiros não-negativos. Receba os valores de a e b pela linha de comando.

19) A partir de um vetor de inteiros, apresente algoritmos recursivos para calcular:
a) O elemento máximo do vetor;
b) O elemento mínimo do vetor;
c) a soma dos elementos do vetor;
d) o produto dos elementos do vetor;
e) a média dos elementos do vetor.

20) Escreva um programa para ler valores de um arquivo contendo valores inteiros, realizar a ordenação e salvar o resultado em outro arquivo.