

```
C ex1.c x
C ex1.c > main()
1 #include <stdio.h>
2
3 int main () {
4     int x, y, *p;
5
6     y = 0;
7     p = &y;
8     x = *p;
9     x = 4;
10    (*p)++;
11    --x;
12    (*p) += x;
13
14    printf("x=%d y=%d *p=%d", x, y, *p);
15 }
```

lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2\$ ./ex1  
x=3 y=4 \*p=4lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2\$

1) Seja o trecho de código abaixo, quais serão os valores de x, y e \*p no comando printf?

```
int main (void) {
    int x, y, *p;

    y = 0;
    p = &y;
    x = *p;
    x = 4;
    (*p)++;
    --y;
    (*p) += x;

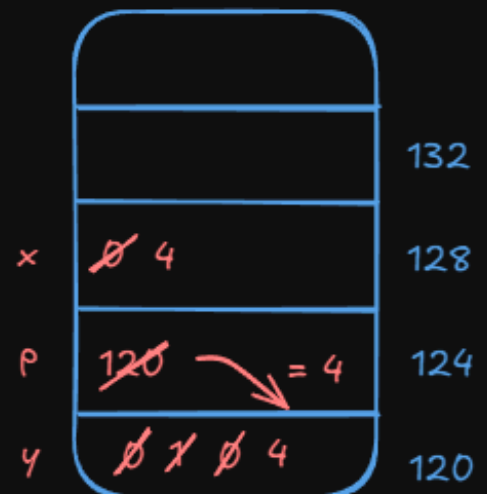
    printf("x=%d y=%d *p=%d", x, y, *p);
}
```

Resposta:

x = 4;

y = 4;

\*p = 4;



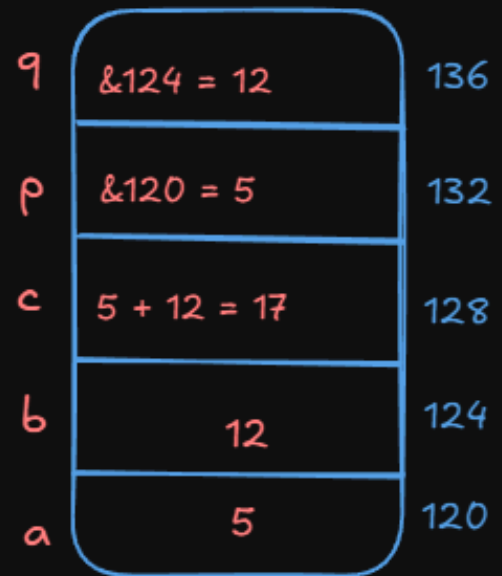
```
C ex2.c x
C ex2.c > ...
1 #include <stdio.h>
2
3 int main (void) {
4     int a=5, b=12, c;
5     int *p;
6     int *q;
7
8     p = &a;
9     q = &b;
10    c = *p + *q;
11
12    printf("c = %d", c);
13 }
```

lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2\$ ./ex2  
c = 17lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2\$

2) Seja o trecho de código abaixo, que valor de c será impresso no comando printf?

```
int main (void) {  
    int a=5, b=12, c;  
    int *p;  
    int *q;  
  
    p = &a;  
    q = &b;  
    c = *p + *q;  
  
    printf("c = %d", c);  
}
```

Resposta:  
c = 17;



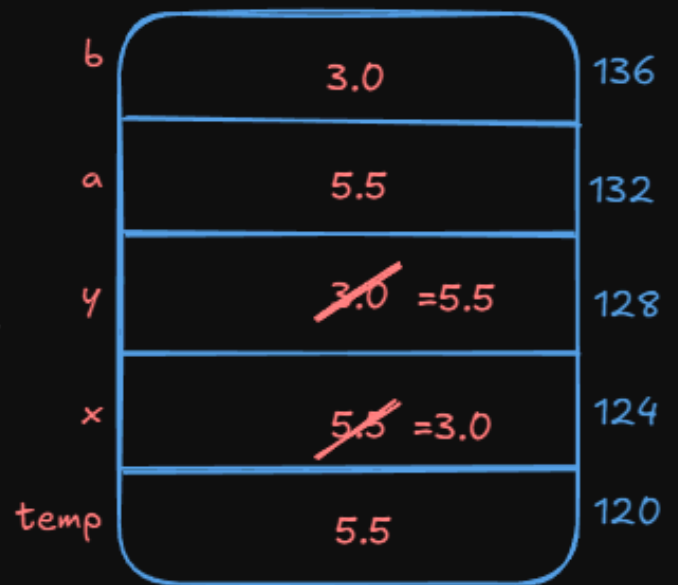
```
C ex3.c X
C ex3.c > ...
1  #include <stdio.h>
2
3  void swapValues(double *x, double *y);
4
5  int main (void) {
6      double a = 5.5;
7      double b = 3.0;
8      swapValues(&a, &b);
9
10     printf("a = %.1f, b = %.1f\n", a, b);
11     return 0;
12 }
13
14 void swapValues(double *x, double *y) {
15     double temp = *x;
16     *x = *y;
17     *y = temp;
18 }

lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2$ ./ex3
a = 3.0, b = 5.5
lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2$
```

3) Seja o trecho de código abaixo, o uso do ponteiro está correto? Se sim, o que está sendo passado como parâmetros para a função "swapValues"?

```
int main (void) {
    double a = 5.5;
    double b = 3.0;
    swapValues(&a, &b);
    return 0;
}

void swapValues(double *x, double *y) {
    double temp = *x;
    *x = *y;
    *y = temp;
}
```



O uso do ponteiro está correto!

O que está sendo passado como parâmetro para função swapValues é o endereço de memória de 'a' e de 'b', que respectivamente está atribuído a 'x' e 'y'. Sendo: a = 3.0 e b = 5.5

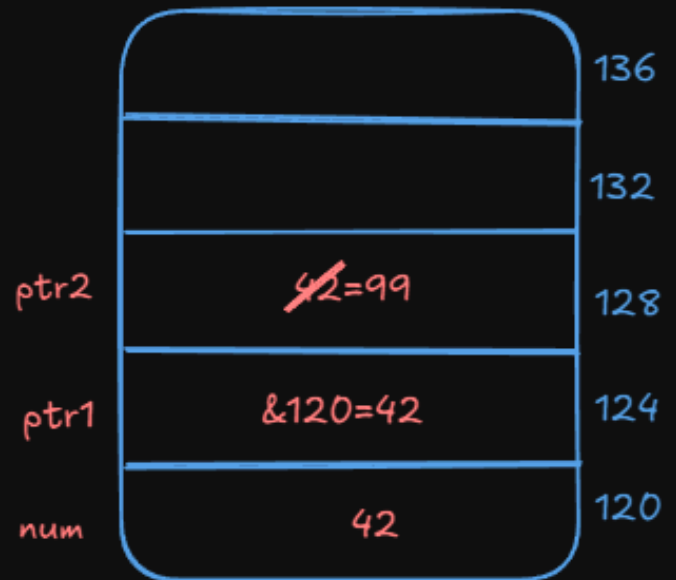
```
C ex4.c X
C ex4.c > ...
1  #include <stdio.h>
2
3  int main (void) {
4      int num = 42;
5      int *ptr1 = &num;
6      int *ptr2 = ptr1;
7      *ptr2 = 99;
8
9      printf("num = %d\n", num);
10     return 0;
11 }
```

lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2\$ ./ex4  
num = 99  
lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2\$

4) Após a execução do Código abaixo, qual é o valor da variável "num"?

```
int main (void) {  
    int num = 42;  
    int *ptr1 = &num;  
    int *ptr2 = ptr1;  
    *ptr2 = 99;  
    return 0;  
}
```

O valor da variável 'num' será de 99.  
num = 99



```
C ex5.c x
C ex5.c > ...
1  #include <stdio.h>
2
3  int main (void) {
4      int x, y, *p;
5      y=0;
6      p=&y;
7      x=*p;
8      x=4;
9      (*p)++;
10     --x;
11     (*p) += x;
12
13     printf("x=%d\n y=%d\n *p=%d\n", x, y, *p);
14 }
```

```
lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2$ ./ex5
x=3
y=4
*p=4
lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2$
```

5) Considerando o código abaixo, indique os valores de x, y e \*p bem como a sua pilha de execução:

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

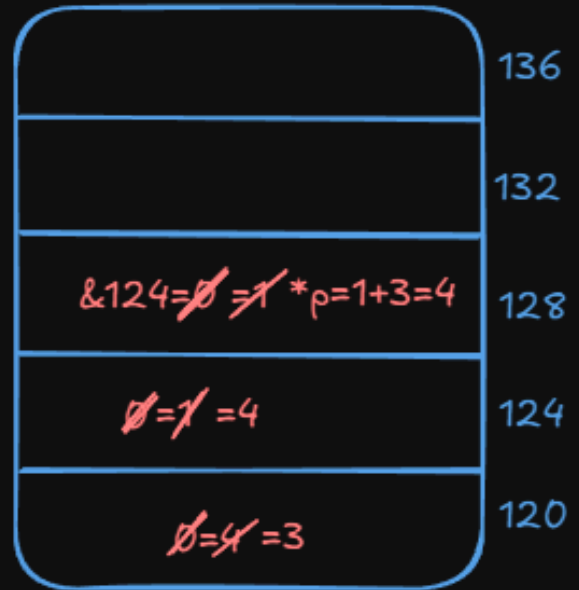
\*p = 1 + 3;  
\*p = 4;

Resposta:  
x = 3;  
y = 4;  
\*p = 4;

p

y

x



```
C ex6.c x
C ex6.c > ...
1 #include <stdio.h>
2
3 int main (void) {
4     int a, b, *p1, *p2;
5     a = 4;
6     b = 3;
7     p1 = &a;
8     p2 = p1;
9
10    *p2 = *p1 + 3;
11    b = b * (*p1);
12    (*p2)++;
13    p1 = &b;
14
15    printf("a = %d\nb = %d\np1 = %d\np2 = %d\n", a,
16 }
```

```
lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2$ ./ex6
a = 8
b = 21
p1 = 21
p2 = 8
lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2$
```

6) Observe o seguinte trecho de Código e indique o valor de cada variável em cada linha:

```
int main (void) {
    a = 4;
    b = 3;
    p1 = &a;
    p2 = p1;
    *p2 = *p1 + 3;
    b = b * (*p1);
    (*p2)++;
    p1 = &b;
}
```

Resposta:

a = 8

b = 21

p1 = 21

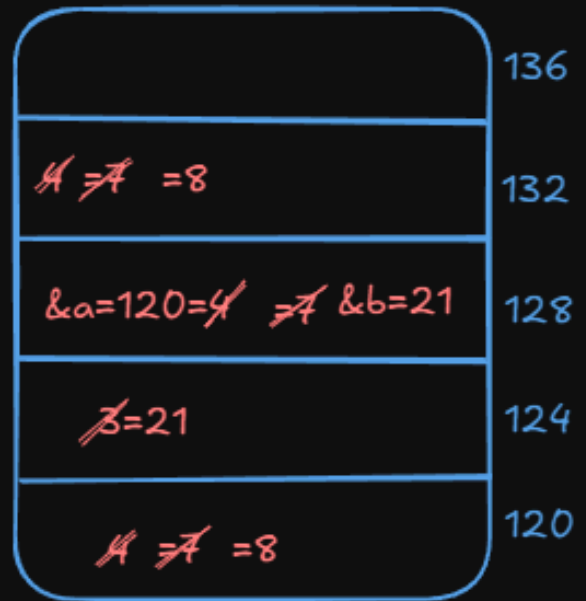
p2 = 8

p2

p1

b

a



```
C ex7.c x
C ex7.c > ...
1 /*
2 7) Escreva um programa que contenha duas variáveis
3 seguidas, compare seus endereços e exiba o conteúdo
4 */
5
6 #include <stdio.h>
7
8 int main (void) {
9     int a, b;
10
11     printf("Variável A e B: ");
12     scanf("%d %d", &a, &b);
13
14     if (&a > &b) {
15         printf("\nConteúdo de maior endereço: %d",
16     } else {
17         printf("\nConteúdo de maior endereço: %d",
18     }
19
20     return 0;
21 }
```

```
lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2$ ./ex7
Variável A e B: 5 4
Conteúdo de maior endereço: 4lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dad
$
```

7) Escreva um programa que contenha duas variáveis inteiras. Leia essas variáveis do teclado. Em seguida, compare seus endereços e exiba o conteúdo do maior endereço.

```
int main (void) {
    int a, b;

    printf("Variável A e B");
    scanf("%d %d", &a, &b);

    if (&a > &b) {
        printf("\nConteúdo de maior endereço: %d", a);
    } else {
        printf("\nConteúdo de maior endereço: %d", b);
    }

    return 0;
}
```

```
C ex8.c x
C ex8.c > ...
1  /*
2  8) Elaborar um programa que leia dois valores inteiros e calcule a soma do dobro dos dois números lidos. A função de soma do dobro de A na própria variável A e o dobro de B na própria variável B.
3  */
4  /*
5  */
6
7  #include <stdio.h>
8
9  int somaDobro(int *a, int *b) {
10     *a = *a * 2; // Dobra A
11     *b = *b * 2; // Dobra B
12     return *a + *b; // Retorna a soma dos valores dobrados
13 }
14
15 int main (void) {
16     int a, b, soma;
17
18     printf("Valor de A: ");
19     scanf("%d", &a);
20     printf("Valor de B: ");
21     scanf("%d", &b);
22
23     //soma = somaDobro(&a, &b);
24
25     printf("\nSoma do dobro: %d\n", soma);
26     printf("A dobrado: %d\n", a);
27     printf("B dobrado: %d\n", b);
28
29     printf("\n\n\n");
30     return 0;
31 }
```

```
Lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2$ ./ex8
Valor de A: 5
Valor de B: 4

Soma do dobro: 32766
A dobrado: 5
B dobrado: 4

Lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2$
```

8) Elaborar um programa que leia dois valores inteiros (A e B). Em seguida faça uma função que retorne a soma do dobro dos dois números lidos. A função deverá armazenar o dobro de A na própria variável A e o dobro de B na própria variável B.

```
int somaDobro(int *a, int *b) {
    *a = *a * 2; // Dobra A
    *b = *b * 2; // Dobra B
    return *a + *b; // Retorna a soma dos valores dobrados
}

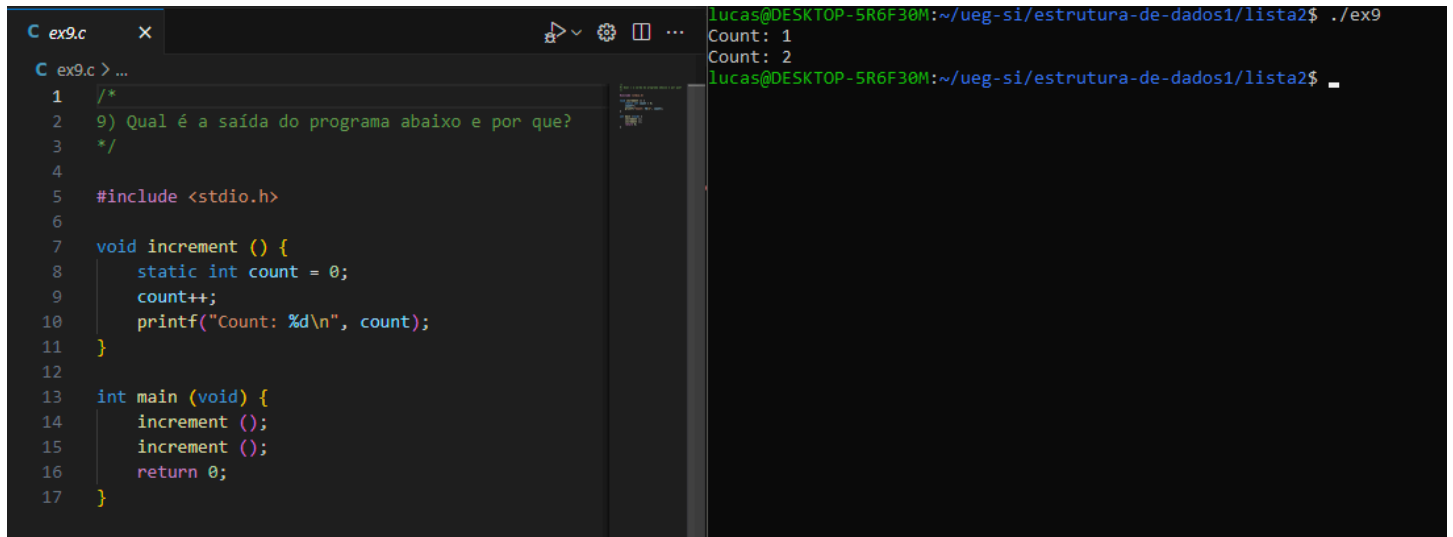
int main (void) {
    int a, b, soma;

    printf("Valor de A: ");
    scanf("%d", &a);
    printf("Valor de B: ");
    scanf("%d", &b);

    soma = somaDobro(&a, &b);

    printf("\nSoma do dobro: %d\n", soma);
    printf("A dobrado: %d\n", a);
    printf("B dobrado: %d\n", b);

    printf("\n\n");
    return 0;
}
```



The screenshot shows a code editor with a C program named `ex9.c` and a terminal window. The code in the editor is as follows:

```
1  /*
2  9) Qual é a saída do programa abaixo e por que?
3  */
4
5  #include <stdio.h>
6
7  void increment () {
8      static int count = 0;
9      count++;
10     printf("Count: %d\n", count);
11 }
12
13 int main (void) {
14     increment ();
15     increment ();
16     return 0;
17 }
```

The terminal window shows the execution of the program. The prompt is `lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2$ ./ex9`. The output is:

```
Count: 1
Count: 2
lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2$ _
```

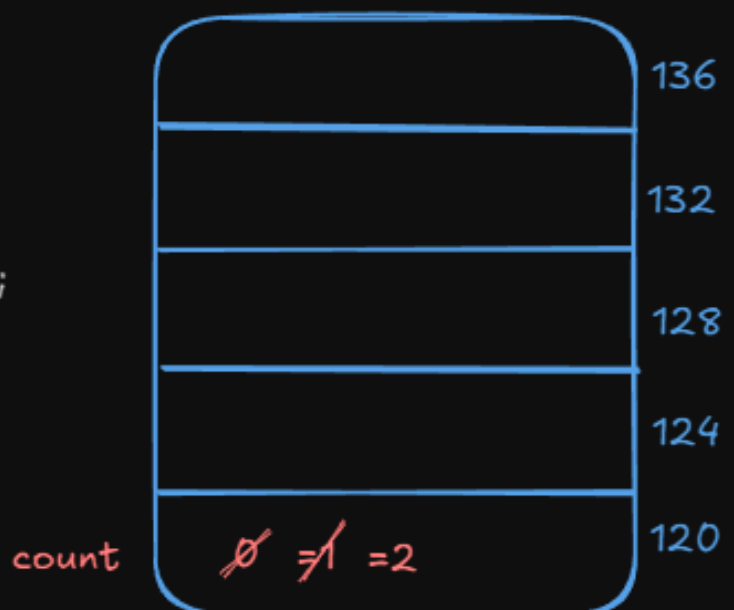


9) Qual é a saída do programa abaixo e por que?

```
#include <stdio.h>

void increment () {
    static int count = 0;
    count++;
    printf("Count: %d\n", count);
}

int main (void) {
    increment ();
    increment ();
    return 0;
}
```



A saída da variável 'count' é 1 e 2, pois no programa principal, a função increment() é chamada, e dentro da função increment() count que recebe o valor 0, é incrementado por count++ (ou count = count+1 -> 0=0+1 -> count=1) e novamente no programa principal a função count é chamada novamente gerando um efeito semelhante a um laço de repetição. Já que o tipo de dado static int mantém seu valor entre chamadas da função. Onde count devido ao último laço que passou de 0 para 1, agora é incrementado de 1 para 2.

```
C ex10.c x
C ex10.c > ...
1  #include <stdio.h>
2
3  int globalVar = 5; // Variável global
4
5  void alteraGlobal() {
6      globalVar = 10; // Modificando a variável global
7  }
8
9  int main() {
10     printf("Antes: %d\n", globalVar);
11     alteraGlobal();
12     printf("Depois: %d\n", globalVar); // Mostra que a variável global foi modificada
13     return 0;
14 }
15

lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2$ ./ex10
Antes: 5
Depois: 10
lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2$
```

10) Qual é a diferença entre uma variável global e uma variável local em C?

- a) Variáveis globais não podem ser modificadas após a declaração.
- b) Variáveis locais não têm um valor inicial definido.
- c) Variáveis locais só podem ser usadas dentro de funções.
- X** d) Variáveis globais podem ser acessadas e modificadas por qualquer função no programa.

```
C ex11.c x
C ex11.c > ...
1  #include <stdio.h>
2
3  typedef struct {
4      char nome[50];
5      int idade;
6      float altura;
7  } Pessoa;
8
9  int main() {
10     Pessoa p; // Declarando uma variável do tipo P
11
12     // Solicitando os dados ao usuário
13     printf("Digite o nome: ");
14     fgets(p.nome, 50, stdin); // Lê uma string com
15
16     printf("Digite a idade: ");
17     scanf("%d", &p.idade);
18
19     printf("Digite a altura: ");
20     scanf("%f", &p.altura);
21
22     // Exibindo os dados inseridos
23     printf("\nDados inseridos:\n");
24     printf("Nome: %s", p.nome); // Não precisa de
25     printf("Idade: %d anos\n", p.idade);
26     printf("Altura: %.2f metros\n", p.altura);
27
28     return 0;
29 }
30
```

```
lucas@DESKTOP-SR6F30M:~/ueg-si/estrutura-de-dados1/lista2$ ./ex11
Digite o nome: Lucas Eduardo
Digite a idade: 26
Digite a altura: 1.79

Dados inseridos:
Nome: Lucas Eduardo
Idade: 26 anos
Altura: 1.79 metros
lucas@DESKTOP-SR6F30M:~/ueg-si/estrutura-de-dados1/lista2$
```

11. Crie uma struct chamada Pessoa, que tenha os seguintes campos:

- nome (string de até 50 caracteres)
- idade (inteiro)
- altura (float)

No programa principal:

1. Declare uma variável do tipo Pessoa.
2. Solicite ao usuário que insira os valores para os campos.
3. Exiba os dados inseridos na tela.

```
typedef struct {
    char nome[50];
    int idade;
    float altura;
} Pessoa;

int main (void) {
    Pessoa p;

    printf("Digite o nome ");
    fgets(p.nome, 50, stdin);

    printf("Digite a idade: ");
    scanf("%i", &p.idade);

    printf("Digite a altura: ");
    scanf("%f", &p.altura);

    printf("\nDados inseridos:\n");
    printf("Nome: %s", p.nome);
    printf("Idade: %d anos\n", p.idade);
    printf("Altura: %.2f metros\n", p.altura);

}
```

C ex12.c x

C ex12.c > ...

```
1  #include <stdio.h>
2
3  typedef struct {
4      char nome[50];
5      int matricula;
6      float nota;
7  } Aluno;
8
9  void exibirAlunos(Aluno alunos[], int tamanho) {
10     printf("\nLista de Alunos:\n");
11     for (int i = 0; i < tamanho; i++) {
12         printf("Aluno %d:\n", i + 1);
13         printf("Nome: %s", alunos[i].nome);
14         printf("Matrícula: %d\n", alunos[i].matricula);
15         printf("Nota: %.2f\n", alunos[i].nota);
16         printf("-----\n");
17     }
18 }
19
20 int main() {
21     Aluno alunos[3]; // Criando um array para 3 alunos
22
23     // Coletando os dados dos alunos
24     for (int i = 0; i < 3; i++) {
25         printf("Digite o nome do aluno %d: ", i + 1);
26         fgets(alunos[i].nome, 50, stdin); // Lê o nome
27
28         printf("Digite a matrícula do aluno %d: ", i + 1);
29         scanf("%d", &alunos[i].matricula);
30
31         printf("Digite a nota do aluno %d: ", i + 1);
32         scanf("%f", &alunos[i].nota);
33
34         getchar(); // Limpa o buffer do teclado
35         printf("\n\n");
36     }
37
38     // Chamando a função para exibir os alunos
39     exibirAlunos(alunos, 3);
40
41     return 0;
42 }
43
```

lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2\$ ./ex12

```
Digite o nome do aluno 1: Lucas
Digite a matrícula do aluno 1: 001
Digite a nota do aluno 1: 8.0
Digite o nome do aluno 2: Eduardo
Digite a matrícula do aluno 2: 002
Digite a nota do aluno 2: 9.0
Digite o nome do aluno 3: Cassiano
Digite a matrícula do aluno 3: 003
Digite a nota do aluno 3: 10.0
```

Lista de Alunos:

Aluno 1:

Nome: Lucas

Matrícula: 1

Nota: 8.00

-----

Aluno 2:

Nome: Eduardo

Matrícula: 2

Nota: 9.00

-----

Aluno 3:

Nome: Cassiano

Matrícula: 3

Nota: 10.00

-----

lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2\$

12. Crie uma struct chamada *Aluno*, que contenha:

- nome (string de até 50 caracteres)
- matricula (inteiro)
- nota (float)

O programa deve:

1. Criar um array para armazenar 3 alunos.
2. Pedir ao usuário para inserir os dados de cada aluno.
3. Criar uma função chamada *exibirAlunos()* que recebe o array e imprime as informações dos alunos formatadas.

```
typedef struct {
    char nome[50];
    int matricula;
    float nota;
} Pessoa;

void exibirAlunos(Aluno alunos[], int tamanho) {
    printf("\nLista de Alunos:\n");
    for (int i = 0; i < tamanho; i++) {
        printf("Aluno %d:\n", i + 1);
        printf("Nome: %s", alunos[i].nome);
        printf("Matrícula: %d\n", alunos[i].matricula);
        printf("Nota: %.2f\n", alunos[i].nota);
        printf("-----\n");
    }
}

int main() {
    Aluno alunos[3];

    for (int i = 0; i < 3; i++) {
        printf("Digite o nome do aluno %d: ", i + 1);
        fgets(alunos[i].nome, 50, stdin); // Lê o nome (com espaços)

        printf("Digite a matrícula do aluno %d: ", i + 1);
        scanf("%d", &alunos[i].matricula);

        printf("Digite a nota do aluno %d: ", i + 1);
        scanf("%f", &alunos[i].nota);

        getchar(); // Limpa o buffer do teclado
    }

    exibirAlunos(alunos, 3);

    return 0;
}
```

C ex12.c

C ex13.c



C ex13.c &gt; ...

```
1  /*
2  13. Crie uma struct chamada Carro, que contenha:
3  • marca (string de até 30 caracteres)
4  • ano (inteiro)
5  • preco (float)
6
7  O programa deve:
8
9  1. Criar uma variável do tipo Carro.
10 2. Criar um ponteiro para essa struct.
113. Pedir ao usuário para preencher os dados.
124. Exibir as informações acessando os valores pelo
13
14 */
15
16 #include <stdio.h>
17
18 typedef struct {
19     char marca[30];
20     int ano;
21     float preco;
22 } Carro;
23
24 int main(void) {
25     Carro carro;           // Variável do tipo Carro
26     Carro *ptrCarro;       // Ponteiro para struct Carro
27     ptrCarro = &carro;    // O ponteiro aponta para o carro
28
29     // Entrada de dados
30     printf("Marca do carro: ");
31     scanf("%s", ptrCarro->marca); // Usando seta para acessar o membro
32
33     printf("Ano de fabricação: ");
34     scanf("%d", &ptrCarro->ano);
35
36     printf("Preço: ");
37     scanf("%f", &ptrCarro->preco);
38
39     // Exibição dos dados
40     printf("\nMarca do carro: %s\n", ptrCarro->marca);
41     printf("Ano de fabricação: %d\n", ptrCarro->ano);
42     printf("Preço: %.2f\n", ptrCarro->preco);
43
44     return 0;
45 }
```

lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2\$ ./ex13

```
Marca do carro: Honda
Ano de fabricação: 2004
Preço: 28000
```

```
Marca do carro: Honda
Ano de fabricação: 2004
Preço: 28000.00
```

lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2\$

13. Crie uma struct chamada Carro, que contenha:

- marca (string de até 30 caracteres)
- ano (inteiro)
- preco (float)

O programa deve:

1. Criar uma variável do tipo Carro.
2. Criar um ponteiro para essa struct.
3. Pedir ao usuário para preencher os dados.
4. Exibir as informações acessando os valores pelo ponteiro.

```
typedef struct {
    char marca[30];
    int ano;
    float preco;
} Carro;

int main(void) {
    Carro carro;
    Carro *ptrCarro;
    ptrCarro = &carro;

    printf("Marca do carro: ");
    scanf("%s", ptrCarro->marca);

    printf("Ano de fabricação: ");
    scanf("%d", &ptrCarro->ano);

    printf("Preço: ");
    scanf("%f", &ptrCarro->preco);

    printf("\nMarca do carro: %s\n", ptrCarro->marca);
    printf("Ano de fabricação: %d\n", ptrCarro->ano);
    printf("Preço: %.2f\n", ptrCarro->preco);

    return 0;
}
```

```
C ex12.c  C ex14.c  x
C ex14.c > Endereco
17 #include <stdio.h>
18
19 typedef struct {
20     char rua[50];
21     int numero;
22     char cidade[30];
23 } Endereco;
24
25 typedef struct {
26     char nome[50];
27     int idade;
28     Endereco endereco; // Struct aninhada
29 } Pessoa;
30
31 int main() {
32     Pessoa pessoa;
33
34     // Coletando os dados da pessoa
35     printf("Digite o nome: ");
36     fgets(pessoa.nome, 50, stdin);
37
38     printf("Digite a idade: ");
39     scanf("%d", &pessoa.idade);
40     getchar(); // Limpa o buffer
41
42     // Coletando os dados do endereço
43     printf("Digite a rua: ");
44     fgets(pessoa.endereco.rua, 50, stdin);
45
46     printf("Digite o número da casa: ");
47     scanf("%d", &pessoa.endereco.numero);
48     getchar(); // Limpa o buffer
49
50     printf("Digite a cidade: ");
51     fgets(pessoa.endereco.cidade, 30, stdin);
52
53     // Exibindo os dados inseridos
54     printf("\nDados da Pessoa:\n");
55     printf("Nome: %s", pessoa.nome);
56     printf("Idade: %d\n", pessoa.idade);
57     printf("Endereco: %s, Nº %d, %s",
58           pessoa.endereco.rua,
59           pessoa.endereco.numero,
60           pessoa.endereco.cidade);
61
62     return 0;
63 }
64
C/C++ Configuration

lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2$ ./ex14
Digite o nome: Lucas
Digite a idade: 26
Digite a rua: Rua Presd Linoln
Digite o número da casa: 1
Digite a cidade: Goiânia

Dados da Pessoa:
Nome: Lucas
Idade: 26
Endereco: Rua Presd Linoln
, Nº 1, Goiânia
lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2$
```



14. Crie um programa que utilize structs aninhadas para armazenar informações de um endereço dentro de uma estrutura de Pessoa.

1. Crie uma struct chamada Endereco, contendo:
  - o rua (string de até 50 caracteres)
  - o numero (inteiro)
  - o cidade (string de até 30 caracteres)
2. Crie uma struct chamada Pessoa, contendo:
  - o nome (string de até 50 caracteres)
  - o idade (inteiro)
  - o endereco (struct Endereco)
3. Peça ao usuário para preencher os dados de uma pessoa e seu endereço.

```
#include <stdio.h>

typedef struct {
    char rua[50];
    int numero;
    char cidade[30];
} Endereco;

typedef struct {
    char nome[50];
    int idade;
    Endereco endereco;
} Pessoa;

int main() {
    Pessoa pessoa;

    printf("Digite o nome: ");
    fgets(pessoa.nome, 50, stdin);

    printf("Digite a idade: ");
    scanf("%d", &pessoa.idade);
    getchar();

    printf("Digite a rua: ");
    fgets(pessoa.endereco.rua, 50, stdin);

    printf("Digite o número da casa: ");
    scanf("%d", &pessoa.endereco.numero);
    getchar();

    printf("Digite a cidade: ");
    fgets(pessoa.endereco.cidade, 30, stdin);

    printf("\nDados da Pessoa:\n");
    printf("Nome: %s", pessoa.nome);
    printf("Idade: %d\n", pessoa.idade);
    printf("Endereco: %s, Nº %d, %s",
           pessoa.endereco.rua,
           pessoa.endereco.numero,
           pessoa.endereco.cidade);

    return 0;
}
```

```
C ex12.c  C ex15.c x
C ex15.c > ...
1  #include <stdio.h>
2
3  typedef struct {
4      char nome[50];
5      float preco;
6      int qtd;
7  } Produto;
8
9  // Função para calcular o total
10 float calcularTotal(Produto *produto) {
11     return produto->preco * produto->qtd;
12 }
13
14 int main(void) {
15     Produto produto;
16     float total;
17
18     // Entrada de dados
19     printf("Nome: ");
20     scanf("%s", produto.nome); // Já é um array, não precisa de &
21
22     printf("Preço: ");
23     scanf("%f", &produto.preco);
24
25     printf("Quantidade: ");
26     scanf("%d", &produto.qtd);
27
28     // Calcula o total chamando a função
29     total = calcularTotal(&produto);
30
31     // Exibe o total
32     printf("Total: %.2f\n", total);
33
34     return 0;
35 }
36

lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2$ ./ex15
Nome: Banana
Preço: 6
Quantidade: 200
Total: 1200.00
lucas@DESKTOP-5R6F30M:~/ueg-si/estrutura-de-dados1/lista2$
```

15. Crie uma struct chamada Produto, que contenha:

- nome (string de até 50 caracteres)
- preco (float)
- quantidade (inteiro)

O programa deve:

1. Criar uma função chamada calcularTotal() que recebe um Produto como parâmetro e retorna o valor total (preco \* quantidade).
2. No main(), pedir ao usuário para inserir os dados de um produto.
3. Exibir o total calculado chamando a função.

```
typedef struct {
    char nome;
    float preco;
    int qtd;
} Produto;

void calcularTotal(Produto *produto, float *preco, int *qtd) {
    return preco * qtd;
}

int main (void) {
    Produto produto;
    float total;

    printf("Nome: ");
    scanf("%s", &produto.nome);
    printf("Preço: ");
    scanf("%f", &produto.preco);
    printf("Quantidade: ");
    scanf("%d", &produto.qtd);

    total = calcularTotal(&produto, &preco, &qtd);

    return 0;
}
```