Daniel da Silva - Curso de Análise e Desenvolvimento de Sistemas (ADS) - UNINOVE - 2025 ©

1. Primeiro Código - Definindo variáveis e exibindo informações

```
#include <stdio.h>

int main() {
    int idade = 30;
    float altura = 1.75;
    double peso = 70.5;
    char inicial = 'A';
    float nota = 7.86;

    printf("Idade: %d\n", idade);
    printf("Altura: %.2f metros\n", altura);
    printf("Peso: %.1f kg\n", peso);
    printf("Inicial do nome: %c\n", inicial);
    nota = 8.5;
    printf("Nota: %.1f\n",nota);
    printf("Voce tem %d anos e sua nota foi %.1f", idade, nota);

    return 0;
}
```

Explicação:

- O código começa definindo variáveis de diferentes tipos: `int`, `float`, `double`, `char`.
- São atribuídos valores a essas variáveis, como `idade = 30`, `altura = 1.75`, etc.
- As variáveis são então impressas usando o `printf()`, com formatação específica para cada tipo:
    - `%d` para inteiros.
    - `%.2f` para números flutuantes (com duas casas decimais).
    - `%.1f` para números flutuantes com uma casa decimal.
    - `%c` para caracteres.

- A nota é alterada para `8.5` e exibida novamente.

Saída esperada:

Idade: 30
Altura: 1.75 metros
Peso: 70.5 kg
Inicial do nome: A
Nota: 8.5
Voce tem 30 anos e sua nota foi 8.5

---

2. Segundo Código - Entrada de idade e nota do usuário

```
#include <stdio.h>
int main() {
    int idade;
    float nota;
    printf("Digite sua idade: ");
    scanf("%d", &idade);

    printf("Digite uma nota: ");
    scanf("%f", &nota);

    printf("Sua idade %d, sua nota: %.1f", idade, nota);
    return 0;
}
```

Explicação:

- O código pede para o usuário digitar sua idade e uma nota.
- A função `scanf()` é usada para ler a entrada do usuário e armazenar os valores nas variáveis `idade` e `nota`.
- Em seguida, os valores inseridos pelo usuário são exibidos com o `printf()`.

Saída esperada:

```
Digite sua idade: 25
Digite uma nota: 8.5
Sua idade 25, sua nota: 8.5
```

---

3. Terceiro Código - Cast de tipos (conversão de tipo)

```
#include <stdio.h>

int main() {
    float num1, num2;
    int x;
    num1 = 7.8;
    x = (int)num1;  // x = 7 (casting de float para int)
    num2 = (float)x; // num2 = 7.0 (casting de int para float)

    printf("Num1 = %.2f \n", num1);
    printf("num2 = %.1f \n", num2);
    printf("x = %d", x);

    return 0;
}
```

Explicação:

- O código realiza operações de conversão (casting) entre tipos de dados:
    - O número `num1` é um `float` (7.8), mas ao ser atribuído à variável `x` (um `int`), ele perde a parte decimal, ficando 7.
    - Depois, a variável `x` é convertida de volta para `float` e atribuída a `num2`, resultando em `7.0`.
- O `printf()` é usado para exibir os valores das variáveis, com as formatações específicas.

Saída esperada:

```
Num1 = 7.80
num2 = 7.0
x = 7
```

---

4. Quarto Código - Operações aritméticas básicas

```
#include <stdio.h>

int main() {
    int primeiroNumero, segundoNumero;
    printf("Digite o primeiro numero: ");
    scanf("%d", &primeiroNumero);
    printf("Digite o segundo numero: ");
    scanf("%d", &segundoNumero);

    printf("Soma: %d \n", primeiroNumero + segundoNumero);
    printf("Subtracao: %d \n", primeiroNumero - segundoNumero);
    printf("Multiplicacao: %d \n", primeiroNumero * segundoNumero);
    printf("Divisao: %.2f \n", (float)segundoNumero / (float)primeiroNumero);
    printf("resto: %d \n", primeiroNumero % segundoNumero);

    return 0;
}
```

Explicação:

- O código pede dois números inteiros e realiza as operações aritméticas básicas: soma, subtração, multiplicação, divisão e resto (módulo).
- A divisão é feita de forma que os números sejam tratados como `float`, para que o resultado seja preciso, incluindo casas decimais.
- O operador `%` calcula o resto da divisão entre os dois números.

Saída esperada:

```
Digite o primeiro numero: 5
Digite o segundo numero: 2
Soma: 7
Subtracao: 3
Multiplicacao: 10
Divisao: 0.40
resto: 1
```

---

5. Quinto Código - Cálculo da média de 3 notas

```
#include <stdio.h>

int main() {
    float n1, n2, n3, media;
    printf("Digite a nota de n1: ");
    scanf("%f", &n1);
    printf("Digite a nota de n2: ");
    scanf("%f", &n2);
    printf("Digite a nota de n3: ");
    scanf("%f", &n3);

    printf("\nNota1: %.1f", n1);
    printf("\nNota2: %.1f", n2);
    printf("\nNota3: %.1f", n3);

    media = (n1 + n2 + n3) / 3;

    printf("\nMedia: %.1f", media);
    printf("\nMedia: %.1f", (n1 + n2 + n3) / 3);

    return 0;
}
```

Explicação:

- O código solicita ao usuário três notas e calcula a média delas.
- A média é obtida somando as três notas e dividindo o resultado por 3.
- As notas e a média são exibidas com uma casa decimal.

Saída esperada:

```
Digite a nota de n1: 7
Digite a nota de n2: 8
Digite a nota de n3: 9
Nota1: 7.0
Nota2: 8.0
Nota3: 9.0
Media: 8.0
Media: 8.0
```

---

6. Sexto Código - Operadores compostos

```
#include <stdio.h>

int main() {
   int a, b, c, d;
   a = 10;
   b = 5;
   a += 2;  // a = a + 2 => a = 12
   b -= 7;  // b = b - 7 => b = -2
   printf("a=%d, b=%d", a, b);

   c = 15;
   d = 2;
   c %= d;  // c = c % d => c = 15 % 2 = 1
   d += a;  // d = d + a => d = 2 + 12 = 14
   printf("\nc=%d, d=%d", c, d);

   return 0;
}
```

Explicação:

- O código utiliza operadores compostos (`+=`, `-=`, `%=`) para modificar as variáveis.
- O operador `%` calcula o resto da divisão (operador módulo).
- O código faz as operações com as variáveis `a`, `b`, `c` e `d` e imprime os resultados.

Saída esperada:

```
a=12, b=-2
c=1, d=14
```

---

7. Operadores Lógicos e Relacionais

```
#include <stdio.h>

int main() {
  int a = 2, b = 3;
  printf("Resultado: %d \n", !(a > b));  // 1
  printf("Resultado: %d \n", (a < b) && !(b <= 1));  // 0
  printf("Resultado: %d \n", !(a > b) || (a <= 7));  // 1

  return 0;
}
```

Explicação:

- O código utiliza operadores lógicos: `!`, `&&` e `||`.
    - `!` é o operador de negação (inverte o valor booleano).
    - `&&` é o operador "E" lógico (retorna verdadeiro apenas se ambas as condições forem verdadeiras).
    - `||` é o operador "OU" lógico (retorna verdadeiro se pelo menos uma das condições for verdadeira).
- O programa realiza comparações e aplica esses operadores para gerar os resultados.
    - `!(a > b)` vai resultar em `1` porque a comparação `a > b` (2 > 3) é falsa, e a negação disso resulta em verdadeiro (1).
    - `(a < b) && !(b <= 1)` vai resultar em `0`, porque `!(b <= 1)` é falso, e qualquer operação com `&&` envolvendo falso resulta em falso.
    - `!(a > b) || (a <= 7)` vai resultar em `1`, porque a negação de `a > b` é verdadeira, e com o `||`, não importa o que aconteça com a segunda parte.

Saída esperada:

```
Resultado: 1
Resultado: 0
Resultado: 1
```

---

8. Operador Ternário (Condicional)

```
#include <stdio.h>

int main() {
  int n, x;
  char r;
  printf("Digite um número: ");
  scanf("%d", &n);

  x = (n % 2 == 0) ? 1 : 0;  // Se n for par, x = 1, caso contrário x = 0
  r = (n % 2 == 0) ? 'P' : 'I';  // Se n for par, r = 'P', caso contrário r = 'I'

  printf("O número %d é %d", n, x);
  printf("\nO número %d é %c", n, r);

  return 0;
}
```

Explicação:

- O operador ternário é uma forma compacta de um `if-else`.
- Aqui, se o número `n` for par (verificado com `n % 2 == 0`), então:
    - `x` recebe `1` e `r` recebe `'P'` (de "par").
    - Caso contrário, `x` recebe `0` e `r` recebe `'I'` (de "ímpar").
- O `printf` exibe o número e os resultados da verificação.

Saída esperada (para n = 5):

O número 5 é 0
O número 5 é I

---

9. Operações de Incremento e Decremento

```
#include <stdio.h>

int main() {
  int i;
  printf("Digite um número: ");
  scanf("%d", &i);

  printf("\n O valor de i = %d", i);  // Exibe o valor original
  i++;  // Incrementa i (i = i + 1)
  i++;
  i++;
  printf("\n O valor de i = %d", i);  // Exibe o valor após incrementos

  i--;  // Decrementa i (i = i - 1)
  i--;
  printf("\n O valor de i = %d", i);  // Exibe o valor após decrementos

  return 0;
}
```

Explicação:

- O código lê um número, incrementa e decrementa o valor da variável `i`.
- O operador `++` é usado para incrementar a variável (adicionando 1) e `--` para decrementar (subtraindo 1).
- O código exibe o valor de `i` antes e depois dos incrementos e decrementos.

Saída esperada (para i = 10):

O valor de i = 10
O valor de i = 13
O valor de i = 11

---

10. Operador de Atribuição Composta

```
#include <stdio.h>

int main() {
  int i;
  printf("Digite um número: ");
  scanf("%d", &i);

  printf("\n O valor de i = %d", i);
  i += 3;  // Aumenta i em 3
  printf("\n O valor de i = %d", i);  // Exibe o valor após incremento

  i--;  // Decrementa 1
  i--;
  printf("\n O valor de i = %d", i);  // Exibe o valor após decrementos

  return 0;
}
```

Explicação:

- O código lê um número e faz o incremento de 3 unidades com o operador `+=`.
- Depois, decrementa o valor de `i` duas vezes com o operador `--`.
- Exibe o valor de `i` após cada operação.

Saída esperada (para i = 10):

O valor de i = 10
O valor de i = 13
O valor de i = 11

---

11. Troca de Valores entre Variáveis

```
#include <stdio.h>

int main() {
  int a, b, aux;
  printf("Digite um número para a: ");
  scanf("%d", &a);
  printf("Digite um número para b: ");
  scanf("%d", &b);

  printf("\n a=%d , b=%d", a, b);  // Exibe os valores originais
  aux = a;  // Guarda o valor de a
  a = b;  // Atribui o valor de b para a
  b = aux;  // Atribui o valor guardado de a para b
  printf("\n a=%d , b=%d", a, b);  // Exibe os valores trocados

  return 0;
}
```

Explicação:

- O código troca os valores de `a` e `b` usando uma variável auxiliar (`aux`).
- Primeiro, os valores de `a` e `b` são exibidos.
- Depois, `a` recebe o valor de `b` e `b` recebe o valor original de `a`, fazendo a troca.

Saída esperada (para a = 7 e b = 5):

```
a=7 , b=5
a=5 , b=7
```

---

12. Condição de Aprovação, Recuperação ou Reprovação

```
#include <stdio.h>

int main() {
  float av1, av2, media;
  printf("Digite uma nota para av1: ");
  scanf("%f", &av1);
  printf("Digite uma nota para av2: ");
  scanf("%f", &av2);

  media = (av1 + av2) / 2;
  if (media >= 6) {
    printf("Media: %.2f, Aprovado", media);
  } else if (media >= 4) {
    printf("Media: %.2f, Recuperação", media);
  } else {
    printf("Media: %.2f, Reprovado", media);
  }

  return 0;
}
```

Explicação:

- O código calcula a média de duas notas e utiliza um `if-else` para determinar a situação do aluno:
    - Se a média for maior ou igual a 6, o aluno está aprovado.
    - Se for maior ou igual a 4 e menor que 6, está em recuperação.
    - Se for menor que 4, o aluno está reprovado.

Saída esperada (para av1 = 7 e av2 = 5):

Media: 6.00, Aprovado

---

13. Switch Case para Localização

```
#include <stdio.h>

int main() {
  int local;
  printf("Digite um número de localização entre 1 e 10: ");
  scanf("%d", &local);

  switch (local) {
    case 1:
      printf("%d - Sul", local);
      break;
    case 2:
      printf("%d - Norte", local);
      break;
    case 3:
      printf("%d - Leste", local);
      break;
    case 4:
      printf("%d - Oeste", local);
      break;
    case 5:
    case 6:
      printf("%d - Nordeste", local);
      break;
    case 7:
    case 8:
    case 9:
      printf("%d - Sudeste", local);
      break;
    case 10:
      printf("%d - Centro Oeste", local);
      break;
    default:
      printf("%d - número desconhecido", local);
  }
  printf("\nOlá mundo!!");

  return 0;
}
```

Explicação:

- O código usa o `switch` para verificar a localização baseada em um número de 1 a 10.

- Dependendo do número inserido, ele exibe o nome da região correspondente.
- Se o número não corresponder a nenhum dos casos, é exibida uma mensagem de erro.

Saída esperada (para local = 5):

5 - Nordeste
Olá mundo!!

14º Código: Estrutura de repetição `while`

```
#include <stdio.h>

int main(){

    int i;

    i = 0;

    while (i <= 10) {

        printf(" %d ", i);

        i += 2;

    }

    return 0;

}
```

Explicação:

- Esse código usa um `while` loop para imprimir números de 0 a 10, saltando de 2 em 2.
- O valor inicial de `i` é 0.
- O `while` continua executando enquanto `i <= 10`. A cada iteração, o valor de `i` é incrementado de 2 (`i += 2`), então os números impressos serão: `0 2 4 6 8 10`.

15º Código: Estrutura de repetição `do while`

```
#include <stdio.h>

int main(){

    int i;

    i = 10;

    do {

        printf(" %d ", i);

        i++;  // incrementa o valor de i

    } while (i < 5);

    return 0;

}
```

Explicação:

- Esse código usa um `do while`. Ele imprime o valor de `i` antes de verificar a condição, por isso sempre imprime o valor inicial de `i` (10) uma vez, independentemente da condição.

- O valor de `i` vai ser incrementado, mas como a condição (`i < 5`) não é mais verdadeira após a primeira execução (pois `i` já é 11), o loop termina após a primeira iteração.

16º Código: Estrutura de repetição `for`

```
#include <stdio.h>

int main(){

    int i;

    for(i = 0; i < 5; i++) {

        printf(" %d ", i);

    }

    printf("\n O valor que tornou a condição falsa: %d", i);

    return 0;

}
```

Explicação:

- Esse código usa um loop `for`, onde o valor de `i` começa de 0 e vai até 4 (`i < 5`).
- O loop imprime os valores de `i`: `0 1 2 3 4`.
- Após o loop, é impresso o valor de `i` quando a condição se torna falsa, que é `5` (porque o loop termina quando `i` chega a 5).

17º Código: Looping Encadeado

```
#include <stdio.h>

int main(){

    int i, j;

    for(i = 10; i < 12; i++) {

        for(j = 2; j < 4; j++) {

            printf("\n%d %d", i, j);

        }

    }

    return 0;

}
```

Explicação:

- Esse código usa loops encadeados. O loop externo (`i`) vai de 10 a 11, e o loop interno (`j`) vai de 2 a 3.

Para cada valor de `i`, o loop interno imprime os valores de `j`, resultando na impressão:

10 2

10 3

11 2

11 3

-

18º Código: Transformar um código em vetor

```
#include <stdio.h>

int main() {

    int x[4];

    x[0] = 1;

    x[1] = 2;

    x[2] = 5;

    x[3] = 3;

    printf("\n x[0] = %d", x[0]);

    printf("\n x[1] = %d", x[1]);

    printf("\n x[2] = %d", x[2]);

    printf("\n x[3] = %d", x[3]);

    return 0;

}
```

Explicação:

- O código original atribui valores a uma variável `x` repetidamente. No entanto, você alterou o código para usar um vetor `x[4]`, onde cada índice do vetor armazena um valor diferente.
- Os valores são atribuídos e impressos: `1 2 5 3`.

19º Código: Usando um loop para imprimir os valores de um vetor

```
#include <stdio.h>

int main() {

    float moeda[5] = {1.00, 0.50, 0.25, 0.10, 0.05};

    int i;

    for(i = 0; i < 5; i++) {

        printf("\n moeda[%d] = %.2f", i, moeda[i]);

    }

    return 0;

}
```

Explicação:

- O vetor `moeda` contém valores monetários. O loop `for` percorre cada índice do vetor e imprime o valor correspondente com duas casas decimais.

A saída será:

moeda[0] = 1.00

moeda[1] = 0.50

moeda[2] = 0.25

moeda[3] = 0.10

moeda[4] = 0.05

-

20º Código: Improvement with a loop and index manipulation

```
#include <stdio.h>

int main() {

    float moeda[5] = {1.00, 0.50, 0.25, 0.10, 0.05};

    int i;

    for(i = 0; i < 5; i += 2) {

        printf("\n moeda[%d] = %.2f", i, moeda[i]);

    }

    return 0;

}
```

Explicação:

- Nesse código, você usou um loop `for` para iterar sobre o vetor de moedas, mas ao invés de percorrer todos os índices, você percorre de 2 em 2 (`i += 2`).
- Ele imprime os valores nos índices 0, 2 e 4: `1.00`, `0.25`, e `0.05`.
