EXERCÍCIOS 3

1. Considere que, no próximo código, a função main chama a função func, passando 5 como argumento. O que será impresso na tela? Desenhe a execução.

func (int n){

if (n == 0)

printf(“fim”);

else

{

printf(n);

func(n-1);

}

}

Análise Inicial: Temos uma função func que recebe um argumento inteiro n. No main(), chamamos func(5).

Chamada Inicial: Quando func(5) é chamada pela primeira vez, n é igual a 5.

Execução de func(5): Como n não é zero, imprimimos o valor de n, que é 5, e então chamamos func(n-1).

Chamada Recursiva para func(4): Agora, func(4) é chamada, e o processo se repete. Imprimimos o valor de n, que é 4, e chamamos func(n-1) novamente.

Recursão Continua: Chamando func(3), func(2), func(1) e finalmente func(0).

Finalização da Recursão: Quando func(0) é chamada, a condição n == 0 é verdadeira, e imprimimos "fim". Em seguida, a execução retorna para a chamada anterior, que é func(1).

Retorno das Chamadas Recursivas: O retorno das chamadas recursivas continua a partir de func(1) para func(2), func(3), func(4), e finalmente func(5).

Resultado Final: A saída impressa no console será a sequência de números de 5 a 1, seguida por "fim". Portanto, a saída será:

5

4

3

2

1

fim

1. Considere que, no próximo código, a função main chama a função func, passando 5 como argumento. O que será impresso na tela? Desenhe a execução.

func (int n){

if (n == 0)

printf(“fim”);

else

{

func(n-1);

printf(n);

}

}

Análise Inicial: Temos uma função func que recebe um argumento inteiro n. No main(), chamamos func(5).

Chamada Inicial: Quando func(5) é chamada pela primeira vez, n é igual a 5.

Execução de func(5): Como n não é zero, chamamos func(n-1), que é func(4), antes de imprimir o valor de n.

Chamada Recursiva para func(4): Agora, func(4) é chamada e a execução continua. Novamente, verificamos se n é igual a zero, o que não é verdadeiro. Chamamos func(n-1), que é func(3), antes de imprimir o valor de n.

Recursão Continua: Repetimos esse processo, chamando func(3), func(2), func(1), e finalmente func(0).

Finalização da Recursão: Quando func(0) é chamada, a condição n == 0 é verdadeira, e imprimimos "fim". Em seguida, a execução retorna para a chamada anterior, que é func(1).

Retorno das Chamadas Recursivas: O retorno das chamadas recursivas continua a partir de func(1) para func(2), func(3), func(4), e finalmente func(5).

Resultado Final: A saída impressa no console será "fim" para func(0) seguido da sequência de números de 1 a 5. Portanto, a saída será:

fim

1

2

3

4

5

1. Considere que, no próximo código, a função main chama a função func, passando 5 como argumento. O que será impresso na tela? Desenhe a execução.

func (int n){

if (n == 0)

printf(“fim”);

else

{

printf(n);

func(n-1);

printf(n);

}

}

Análise Inicial: Temos uma função func que recebe um argumento inteiro n. No main(), chamamos func(5).

Chamada Inicial: Quando func(5) é chamada pela primeira vez, n é igual a 5.

Execução de func(5): Como n não é zero, imprimimos o valor de n, que é 5, e então chamamos func(n-1).

Chamada Recursiva para func(4): Agora, func(4) é chamada, e o processo se repete. Imprimimos o valor de n, que é 4, chamamos func(n-1) e, em seguida, imprimimos novamente o valor de n.

Recursão Continua: Repetimos esse processo, chamando func(3), func(2), func(1) e finalmente func(0).

Finalização da Recursão: Quando func(0) é chamada, a condição n == 0 é verdadeira, e imprimimos "fim". Em seguida, a execução retorna para a chamada anterior, que é func(1).

Retorno das Chamadas Recursivas: O retorno das chamadas recursivas continua a partir de func(1) para func(2), func(3), func(4), e finalmente func(5).

Resultado Final: A saída impressa no console será a sequência de números de 5 a 1, seguida por outra sequência de números de 1 a 5. Portanto, a saída será:

5

4

3

2

1

fim

1

2

3

4

5

1. Considere que, no próximo código, a função main chama a função func, passando 5 como argumento. O que será impresso na tela? Desenhe a execução.

func (int n){

if (n == 0)

printf(“fim”);

else

{

func(n-1);

printf(n);

func(n-1);

}

}

Análise Inicial:Temos uma função func que recebe um argumento inteiro n. No main(), chamamos func(5).

Chamada Inicial: Quando func(5) é chamada pela primeira vez, n é igual a 5.

Execução de func(5): Como n não é zero, a execução continua. Na sequência, chamamos func(n-1), que é func(4).

Chamada Recursiva para func(4): Agora, func(4) é chamada. Continuamos esse processo chamando func(n-1) e, neste caso, será func(3).

Recursão Continua: A recursão continua com chamadas para func(3), func(2), func(1), e finalmente func(0).

Condição de Parada: Quando func(0) é chamada, a condição n == 0 é verdadeira, e imprimimos "fim". Isso acontece para a primeira chamada de func(0).

Retorno das Chamadas Recursivas: A execução retorna para a chamada anterior, que é func(1).

Impressão de n e Chamada Recursiva: Agora, imprimimos o valor de n, que é 1, e depois chamamos func(1-1), o que é func(0) novamente.

Condição de Parada para a Segunda Chamada de func(0): A segunda chamada para func(0) também imprime "fim".

Retorno das Chamadas Recursivas: Agora, a execução retorna para func(1).

Conclusão da Execução de func(1): Após a segunda chamada para func(0), a função func(1) termina.

Retorno das Chamadas Recursivas: Esse processo de retorno continua para func(2), func(3), func(4), e finalmente func(5).

Resultado Final: A saída impressa no console será a sequência "fim" seguida por números de 1 a 5 e novamente a sequência "fim". Portanto, a saída será:

fim

1

fim

2

fim

3

fim

4

fim

5

1. Considere que, no próximo código, a função main chama a função func, passando 5 como argumento. O que será impresso na tela? Desenhe a execução.

func (int n){

if (n == 0)

printf(“fim”);

else

{

printf(n);

func(n-1);

func(n-1);

}

}

Análise Inicial: Temos uma função func que recebe um argumento inteiro n. No main(), chamamos func(5).

Chamada Inicial: Quando func(5) é chamada pela primeira vez, n é igual a 5.

Execução de func(5): Como n não é zero, imprimimos o valor de n, que é 5, e então chamamos func(n - 1) e func(n - 1).

Recursão: Agora, temos duas chamadas recursivas para func(4), uma após a outra.

Chamada 1: func(4)

Verificamos se n é igual a zero. Como n não é zero, imprimimos o valor de n, que é 4, e então chamamos func(n - 1) e func(n - 1) novamente.

Chamada 2: func(4) (outra recursão)

Repetimos o mesmo processo de imprimir 4 e, em seguida, fazer duas chamadas para func(3).

Recursão Continua: O processo de recursão continua a se ramificar e descer pela árvore de chamadas. Cada chamada recursiva imprime o valor de n e faz duas novas chamadas para func(n - 1).

Finalização da Recursão: Quando chegamos a func(0), a condição n == 0 é verdadeira, e imprimimos "fim". No entanto, cada chamada recursiva retorna ao nível anterior da recursão, continuando a imprimir os valores de n e executando chamadas recursivas.

Resultado Final: O resultado impresso no console será uma árvore de chamadas que começa com "5" e se ramifica em múltiplas chamadas recursivas até que todas cheguem a "fim". A sequência exata de saída depende da ordem em que as chamadas recursivas são processadas, mas uma possível saída seria algo como:

5

4

3

2

1

fim

fim

fim

fim

1. Considere que, no próximo código, a função main chama a função func, passando 5 como argumento. O que será impresso na tela? Desenhe a execução.

func (int n){

if (n == 0)

printf(“fim”);

else

{

func(n-1);

func(n-1);

printf(n);

}

}

Análise Inicial: Temos uma função func que recebe um argumento inteiro n. No main(), chamamos func(5).

Chamada Inicial: Quando func(5) é chamada pela primeira vez, n é igual a 5.

Execução de func(5): Como n não é zero, entramos no bloco else.

Chamadas Recursivas para func(n-1): Duas chamadas recursivas são feitas para func(4) (ou seja, func(n-1) duas vezes). Isso significa que a função func será executada duas vezes com n igual a 4.

Execução de func(4): Cada uma das chamadas recursivas para func(4) entra no bloco else novamente e faz mais duas chamadas recursivas para func(3).

Recursão Continua: Esse padrão de chamadas recursivas se repete até que alcancemos func(0).

Impressão de Números: Quando chegamos a func(0), imprimimos "fim". Em seguida, a execução retorna para a chamada anterior, que é func(1).

Retorno das Chamadas Recursivas: As chamadas recursivas continuam a retornar até chegarmos à chamada inicial de func(5).

Resultado Final: A saída impressa no console será a sequência de números de 5 a 1, seguida por "fim", e essa sequência será impressa duas vezes devido às chamadas duplicadas para func(n-1).

5

4

3

2

1

fim

5

4

3

2

1

fim

1. Observe atentamente o código abaixo:

int fazAlgo(int x, int y){

if(y == 0){

return x;

}else{

return fazAlgo(y, x % y);

}

}

int main() {

int a,b,res;

printf("Informe o primeiro inteiro: ");

scanf("%d",&a);

printf("Informe o segundo inteiro: ");

scanf("%d",&b);

res=fazAlgo(a,b);

printf("O resultado dos valores %d e %d=%d",a,b,res);

}

Se o usuário digitar 12 e 18, qual será o resultado? Para verificar o resultado, faça o desenho da execução. Lembre-se, na prova não há computador para ajudar a verificar o resultado.

Se o usuário digitar 6 e 9, qual será o resultado? Para verificar o resultado, faça o desenho da execução.

O que este programa calcula?

Quando o usuário digitar 12 e 18:

Chamada Inicial:

fazAlgo(12, 18) é chamado.

Execução de fazAlgo(12, 18):

Verificamos se y é zero. Como y não é zero, chamamos fazAlgo(18, 12 % 18).

12 % 18 é igual a 12.

Execução de fazAlgo(18, 12):

Verificamos se y é zero. Como y não é zero, chamamos fazAlgo(12, 18 % 12).

18 % 12 é igual a 6.

Execução de fazAlgo(12, 6):

Verificamos se y é zero. Como y não é zero, chamamos fazAlgo(6, 12 % 6).

12 % 6 é igual a 0.

Execução de fazAlgo(6, 0):

Verificamos se y é zero. Como y é zero, retornamos x, que é 6.

Retorno das Chamadas Recursivas:

O retorno da função fazAlgo(6, 0) é 6.

A chamada anterior, fazAlgo(12, 6), recebe esse valor e retorna 6 também.

A chamada inicial, fazAlgo(12, 18), retorna 6.

Resultado: O resultado dos valores 12 e 18 é 6.

Quando o usuário digitar 6 e 9:

Chamada Inicial:

fazAlgo(6, 9) é chamado.

Execução de fazAlgo(6, 9):

Verificamos se y é zero. Como y não é zero, chamamos fazAlgo(9, 6 % 9).

6 % 9 é igual a 6.

Execução de fazAlgo(9, 6):

Verificamos se y é zero. Como y não é zero, chamamos fazAlgo(6, 9 % 6).

9 % 6 é igual a 3.

Execução de fazAlgo(6, 3):

Verificamos se y é zero. Como y não é zero, chamamos fazAlgo(3, 6 % 3).

6 % 3 é igual a 0.

Execução de fazAlgo(3, 0):

Verificamos se y é zero. Como y é zero, retornamos x, que é 3.

Retorno das Chamadas Recursivas:

O retorno da função fazAlgo(3, 0) é 3.

A chamada anterior, fazAlgo(6, 3), recebe esse valor e retorna 3 também.

A chamada inicial, fazAlgo(6, 9), retorna 3.

Resultado: O resultado dos valores 6 e 9 é 3.

Este programa implementa o algoritmo para encontrar o máximo divisor comum (MDC) entre dois números usando o algoritmo de Euclides. A função fazAlgo retorna o máximo divisor comum entre os dois números fornecidos como argumentos.