Algoritmos e Estruturas de Dados I

Prof. Lúcio Mauro Pereira

25/09/2024

Questão: Primos entre si

Construa uma função que verifique se dois números inteiros são primos entre si.

Dois números inteiros são ditos primos entre si caso não exista divisores comuns a ambos, exceto o número 1.

Observe que para dois números serem primos entre si não é necessário serem eles números primos. Ao contrário, a única coisa a ser observada é comportamento de ambos quanto aos seus divisores.

Parâmetros: dois inteiros, relativos aos dois números a serem comparados.

Valor gerado: *verdadeiro*, se os dois números naturais forem primos entre si, ou *falso*, caso contrário.

Obs: se um dos inteiros for o valor zero, a função deverá gerar falso.

Primos entre si

```
bool primosEntreSi(int a, int b) {
    bool primos;









    return primos;
}
```

Primos entre si

```
bool primosEntreSi(int a, int b) {
    bool primos;
    if(b==0)primos=false;
    else {
        primos=true;









    }
    return primos;
}
```

Primos entre si

```
bool primosEntreSi(int a, int b) {
    bool primos;
    if(b==0)primos=false;
    else {
        primos=true;
        int i=2;









    }
    return primos;
}
```

Primos entre si

```
bool primosEntreSi(int a, int b) {
    bool primos;
    if(b==0)primos=false;
    else {
        primos=true;
        int i=2;
        int menor;
        if(a<b)menor=a;
        else    menor=b;




    }
    return primos;
}
```

Primos entre si

```
bool primosEntreSi(int a, int b) {
    bool primos;
    if(b==0)primos=false;
    else {
        primos=true;
        int i=2;
        int menor;
        if(a<b)menor=a;
        else    menor=b
        while(primos && i<=menor) {

            i++;
        }
    }
    return primos;
}
```

Primos entre si

```
bool primosEntreSi(int a, int b) {
    bool primos;
    if(b==0)primos=false;
    else {
        primos=true;
        int i=2;
        int menor;
        if(a<b)menor=a;
        else    menor=b;
        while(primos && i<=menor) {
            if(a%i==0 && b%i==0) primos = false;
            i++;
        }
    }
    return primos;
}
```

Tente

O problema do número primos entre si é um clássico problema para apresentar algoritmos recursivos.

1) Tente construir uma proposta de solução para este problema utilizando a abordagem recursiva

Trata-se de um problema que também permite diversos propostas de soluções, diversos algoritmos distintos. Tente associá-lo ao problema do **MDC** e proponha com ele:

2) Abordagem iterativa

3) Abordagem recursiva

Questão: Série de Fetuccine

A série de FETUCCINE é gerada da seguinte forma: os dois primeiros termos são dados; a partir daí, os termos seguintes são gerados com a soma ou subtração dos dois termos anteriores, seguindo o comportamento abaixo:

$A_i = A_{i-1} + A_{i-2}$, *para i par*

$A_i = A_{i-2} - A_{i-1}$, *para i ímpar*

Em outras palavras, quando em posição par, um novo número é obtido pela soma dos dois anteriores; por outro lado, quando em posição ímpar, um novo número é obtido subtraindo o número anterior do penúltimo.

Exibir na tela do monitor de vídeo os *n* primeiros termos da série.

Planeje, cuidadosamente, os parâmetros a serem encaminhados e o valor gerado pela função. De imediato, note que os dois primeiros termos, bem como o número de termos, precisam ser parametrizados.

(5,0) Abordagem iterativa

(5,0) Abordagem recursiva

Série de Fetuccine

```
void fetuccineIterativo(int a, int b, int n) {
    printf("%i, %i ", a, b);
    int prox;




}
```

Série de Fetuccine

```
void fetuccineIterativo(int a, int b, int n) {
    printf("%i, %i ", a, b);
    int prox;
    for (int i = 3; i <= n; i++) {
        if (i % 2 == 0)   prox = a + b;
        else              prox = b – a;
        printf("%i ", prox);



    }
}
```

Série de Fetuccine

```
void fetuccineIterativo(int a, int b, int n) {
    printf("%i, %i ", a, b);
    int prox;
    for (int i = 3; i <= n; i++) {
        if (i % 2 == 0)   prox = a + b;
        else              prox = b – a;
        printf("%i ", prox);
        a = b;
        b = prox;
    }
}
```

Fetuccine recursivo

```
void fetuccineRecursivo(int a, int b, int n, int i) {
    if (i <= n) {




    }
}
```

Fetuccine recursivo

```
void fetuccineRecursivo(int a, int b, int n, int i) {
    if (i <= n) {
        int prox;
        if (i % 2 == 0)  prox = a + b;
        else prox = b - a;
        printf("%i ", prox);

    }
}
```

Fetuccine recursivo

```
void fetuccineRecursivo(int a, int b, int n, int i) {
    if (i <= n) {
        int prox;
        if (i % 2 == 0)  prox = a + b;
        else prox = b - a;
        printf("%i ", prox);
        fetuccineRecursivo(b, prox, n, i + 1);
    }
}
```

Fetuccine recursivo

```
void fetuccine(int a, int b, int n) {
        printf("%i %i ", a, b);
        fetuccineRecursivo(a, b, n, 3);
}
```

Soma dos números ímpares

Uma função deverá calcular e retornar a soma dos números ímpares comprendidos no intervalo de 1 até *N*, sendo *N* um valor parametrizado.

Planeje, cuidadosamente, os parâmetros a serem encaminhados e o valor gerado pela função.

(5,0) Abordagem iterativa

(5,0) Abordagem recursiva

Alocação dinâmica de memória

Construa uma percepção de como se dá, na memória primária (RAM), a alocação de espaços quando declaramos uma variável.

Alocação dinâmica de memória

A função malloc(?)