Luan Lucas L. Peloso

Universidade Federal da Fronteira Sul - UFFS
Ciência da Computação
Estrutura de Dados

Instruções

1. Coloque apenas a sua MATRÍCULA na folha resposta.
2. Aparelhos eletrônicos desligados (ou no modo silencioso).
3. Responda TODAS questões na folha resposta, enumerando as mesmas na ordem que melhor lhe convier.
4. As questões podem ser resolvidas a lápis porém o professor se reserva a não aceitar reclamações oriundas da correção das questões.
5. Consulta permitida apenas a cola oficial.

Avaliação 01

1. A divisão euclidiana é o processo de dividir um inteiro (dividendo) por outro (divisor) de forma que produza um quociente e um resto menor que o divisor. Sendo assim, dados dois inteiros a e b, com $b \neq 0$, existem dois números únicos q e r tal que $a = bq + r$ e $0 \leq r < |b|$. A divisão Euclidiana é realizada através de subtrações sucessivas. Por exemplo, o resto de 7 por 3 (7%3) é calculado: 7-3, 4-3 a próxima subtração 1-3 não é realizada pois 1 é menor que 3 e maior que zero, neste caso 1 é o resto da divisão de 7 por 3. Baseado nessa explicação, apresente a definição recursiva do problema e em seguida implemente a definição em C.

2. Você foi designado para fazer uma nova TAD de números com ponto flutuante para sua equipe que será utilizada para armazenar distâncias em metros. Basicamente um número de ponto flutuante possui duas partes, uma parte inteira e uma parte decimal (números depois da vírgula). Por exemplo, 8.32, onde 8 é a parte inteira e 32 a parte decimal. Como sua biblioteca será utilizada para armazenar distâncias não devem existir números onde a parte decimal tenha mais de dois dígitos, ou seja, cada vez que a biblioteca realizar operações aritméticas sobre a parte decimal, quando esta passar de 99 é acrescido 1 unidade na parte inteira. Declare um tipo abstrato de dados que represente um número de ponto flutuante que consista em uma struct contendo dois membros, inteiro e decimal, ambos do tipo inteiro, e faça o seguinte:

   (a) Escreva uma função `setNumber` que recebe dois argumentos do tipo inteiro (a parte inteira e a parte decimal do número), os armazena em uma variável do tipo Flutuante e retorna esta variável. Você pode optar por implementar a função na forma `cp=setNumber(10,20)` ou `setNumber(&cp,10,20)`

   (b) Escreva a função sumNumbers que recebe dois argumentos do tipo Flutuante, soma os valores dos membros correspondentes das structs recebidas. Ex. $2,51 + 2,53 = 5,04$

   (c) Escreva uma função printNumeber que recebe um número Flutuante como parâmetro e imprime o número no formato: *x,y metros*, e.g., 2,34 *metros*.

* 3. Um Analista de Sistemas recebeu a tarefa de implementar uma sub-rotina que, a partir de um vetor "A", gerasse um outro vetor de saída "B". Um pedaço desta sub-rotina, implementada em C, está apresentado na Figura 1.

```
int vet[N];
int i,j, aux;
 i =0; j = N-1;
while (i<j){
    aux = vet[i];
    vet[i++] = vet[j];
    vet[j--] = aux;
}
```

Figure 1: Fragmento de código

Considere o vetor $vet = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Baseado no trecho de código da Figura 1, apresente como estará o conteúdo de vet após a exexução do código.

BOA PROVA & BOA SORTE.

Luan Lucas de Lima Peloso

3 vet = {9, 8, 7, 6, 5, 4, 3, 2, 1}

2
```
#include <stdio.h>

typedef struct {
    int inteiro;
    int decimal;
} Flutuante;
```

a
```
Flutuante setNumber (int inteiro, int decimal){
    Flutuante num;
    num.inteiro = inteiro;
    num.decimal = decimal;
    return num;
}
```

b
```
Flutuante sumNumbers (Flutuante n, Flutuante m){
    Flutuante soma;
    if (n.decimal + m.decimal > 100){
        soma.decimal = (n.decimal+m.decimal)%100;
        soma.inteiro = n.inteiro + m.inteiro + (n.decimal+m.decimal)
        /100;
    }else {
        soma.decimal = n.decimal + m.decimal;
        soma.inteiro = n.inteiro + m.inteiro;
    }
    return soma;
}
```

© void printNumber (Flutuante x) {
Printf ("\t%d,%d metros\n", x.inteiro, x.decimal);
return;

int main(){
Flutuante disA, disB, soma;
disA = setNumber (2, 51);
disB = setNumber (2, 53);
soma = sumNumbers (disA, disB);
printNumber (disA);
printNumber (disB);
printNumber (soma);
return 0;
}

-5 ? -7 -3

1 int mod (int a, int b) {
if (b>0) {
if a>0 {
if a<b {
return a
}
return mod (a-b, b)
}
if a<0 {
return mod (a+b, b)
}
}
if (b<0) {
if (a>0) {
if (a>(b*-1)) {

```
        return a;
        }
        return mod (a(-1), b(-1))
        }
    if (a<0){
        return mod (a-b, b)
        }
    }
}
```

Definição: para este problema temos 4 tipos de entrada possíveis (a, b); (a, -b); (-a, b); (-a, -b) então, separamos por partes e vamos resolvendo para cada entrada:

1 (b>0)

1.1 (b>0, a>0)

Caso a e b sejam positivos, podemos seguir com a subtração comum até que a<b, sendo a resto.

1.2 (b>0, a<0)

Caso a seja negativo e b seja positivo, precisamos realizar a soma (-a+b); (b) até que a se torne positivo e encaixe-se no 1.1

2 (b<0)

2.1 (b<0, a>0)

- podemos inverter os sinais de ambos os partes pois $\frac{a}{-b} = \frac{-a}{b}$, encaixando-se no 1.2.
- Criamos também o retorno de a, se for maior que |b|, para ser usado no 2.1

2.2 (b<0, a<0)

Caso ambos sejam negativos, realizamos a

subtração até que a se torne positivo, e encerra-se com o resto a > |b|, criado no 2.1