Instituto Federal de Educação, Ciência e Tecnologia da Paraíba

Campus Campina Grande

Cursos de Engenharia de Computação: **Lab. de Estruturas de Dados e Algoritmos**

Roteiro Prático I - Structs e Alocação Dinâmica de Memória

1. Conceitos Iniciais

Neste exercício vamos praticar os conceitos básicos de struct e alocação dinâmica de memória. Verifique o código definido a seguir e responda as perguntas que vem em seguida:

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

typedef struct {
    int idade;
    char nome[50];
    int *notas;
} aluno;

int main() {
    aluno a1, a2;
    a1.idade = 10;
    strcpy(a1.nome, "Pedro"); //a1.nome = "Pedro" não funciona
    a1.idade = 12;
    strcpy(a1.nome, "Maria");

    a1.notas = (int*)malloc(5*sizeof(int));
    a2.notas = (int*)malloc(4*sizeof(int));

    int i;
    for(i = 0; i < 5; i++) {
        a1.notas[i] = 100;
    }
    a2.notas = a1.notas;
    for(i = 0; i < 4; i++) {
        a2.notas[i] -= 10;
    }
    double media = 0;
    for(i = 0; i < 5; i++) {
        media += a1.notas[i];
    }
    media = media/5;
    printf("Media de %s: %lf\n",a1.nome,media);

    media = 0;
    for(i = 0; i < 4; i++) {
        media += a2.notas[i];
    }
    media = media/4;
    printf("Media de %s: %lf\n",a2.nome,media);

    free(a1.notas);
    free(a2.notas);

    return 0;
}
```

Questão 1: Qual a saída fornecida por esse programa?

Questão 2: O que ocorre na linha 25?

Questão 3: O que ocorre nas linhas 43 e 44? Devido ao que foi feito anteriormente no código, isso pode gerar algum problema?

Questão 4: Reescreva esse código e corrija os erros para que ele funcione da maneira esperada. Ou seja, deve-se colocar corretamente na saída a média de cada aluno, sabendo que as 4 notas do alunos a2 equivalem às 4 primeiras notas do aluno a1 subtraídas de 10.

2. Função splitInt

Neste exercício, você deve implementar uma função semelhante à função split disponível em python. Essa função, denominada *splitInt*, deve receber como parâmetro uma string contendo números inteiros separados por espaços em branco e deve retornar um ponteiro para um array de inteiros construído após a separação dos valores inteiros contidos na string. Por exemplo, se for recebida como parâmetro a string (array de **char**) “3 24 55 2 0”, deve-se gerar um array de inteiros de 5 posições (V = {3, 24, 55, 2, 0}). É importante notar que essa função deve funcionar, mesmo que existam espaços em branco no início ou no final na string, bem como uma quantidade variável de espaços em branco entre os números. Por outro lado, é garantido que todos os números inteiros contidos na string cabem em uma variável do tipo **int**.

2.1 Função para Verificação e Contagem de Números

Antes de implementar a função splitInt, deve-se implementar uma função auxiliar para verificar se a string passada como parâmetro é válida (ou seja, contém apenas números e espaços em branco) e para retornar a quantidade de números independentes contidos na string. Essa função será usada dentro da função splitInt.

A função para verificação e contagem, denominada verifyListInt deve receber como parâ- metro um array de char e deve retornar a quantidade de números independentes contidos na string. Caso não exista nenhum número na string ou se existir algum caractere inválido, deve-se retornar o valor 0.

2.2 Implementação da Função splitInt

A função *splitInt* deve receber como parâmetro um array de char e um valor inteiro **passado por referência**, que receberá o tamanho do array de inteiros que foi gerado a partir da string. A função deve retornar um ***int ****, que deve apontar para um array de inteiro criado dinamicamente a partir dos valores contidos na string.

No início da função, deve-se utilizar a função implementada anteriormente (*verifyListInt*) para obter a quantidade de números contidos na string. Caso a função *verifyListInt* retorne o valor 0, deve-se atribuir o valor 0 ao valor inteiro passado como parâmetro por referência e deve-se retornar NULL.

Caso o valor retornado por *verifyListInt* seja um número maior que zero, deve-se atribuir esse tamanho à variável inteira passada como parâmetro por referência e deve-se alocar um vetor de inteiros utilizando o tamanho retornado por verifyListInt. Após isso, deve-se separar os números contidos na string, convertê-los para inteiro (pode usar a função pronta *atoi* - `stdlib.h`) e colocá-los no array que foi criado dinamicamente. Ao final, deve-se retornar o ponteiro para esse novo array criado dinamicamente.

As assinaturas das duas funções a serem criadas são mostradas a seguir:

```
1     int verifyListInt(const char *s);
2     int* splitInt(const char *s, int *size);
```

O código a seguir mostra a função main() de um programa que testa a função *splitInt*.

```
1  int main() {
2      char ent[100];
3      scanf("%[^\n]s", ent);
4      int size ;
5      int *ent_int = split_int(ent,&size);
6      if(ent_int == NULL) {
7          printf ("String inválida");
8          return 1;
9      }
10     printf("Size: %d\n",size);
11     int i;
12     for(i = 0; i < size; i++) {
13         printf("%d ",ent_int[i]);
14     }
15     return 0 ;
16 }
```