Décima Primeira Lista de Exercícios SSC0502

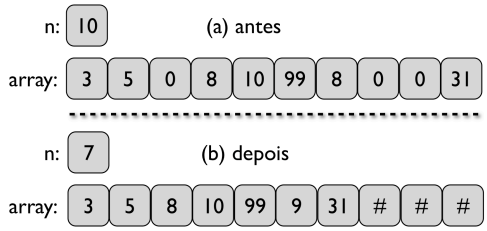
1º Semestre - 9 Junho de 2010 Prof. Márcio Delamaro

Nome: _____ USP: _____

Laboratório de Introdução à Ciência da Computação

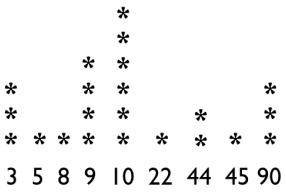
---

1. **Escreva uma função denominada `removeElementosZero(array, n)` que percorre um array de inteiros e elimina os elementos cujos valores são iguais a zero. Visto que essa função altera o tamanho do array, `removeElementosZero` deve retornar o novo tamanho do array alterando o valor da variável `n`. A funcionalidade proposta é exemplificada na figura a seguir. (Crie uma programa para testar essa função.)**



---

2. **Estenda o programa do exercício 1 criando outra função que exiba um histograma dos números presentes no array passado como parâmetro. O histograma deve ser como ilustrado a seguir.**



---

3. **Escreva um programa que informa quando um inteiro digitado pelo usuário é um número feliz (*happy integer*). Um número feliz é determinado por meio do seguinte processo: cada digito do número elevado ao quadrado é novamente elevado ao quadrado e somado, o processo continua até que o resultado seja 1. É importante destacar que o programa também pode entrar em loop infinito, assim o programa deve ser encerrado após 3000 iterações.**

- **Por exemplo, caso o usuário digite o número 7 o processo é como a seguir:**

```
7^2 = 49
4^2 + 9^2 = 97
9^2 + 7^2 = 130
1^2 + 3^2 + 0^2 = 10
1^2 + 0^2 = 1
```

4. **A notação usual para expressões aritméticas é conhecida como notação *infixa*. Utilizando essa notação representa-se uma operação binária na forma `x+y`, ou seja, com o operador entre seus dois operandos. Uma notação alternativa é a notação *posfixa*, também conhecida como notação polonesa, na qual o operador é expresso após seus operandos. O atrativo da notação posfixa é que ela dispensa o uso de parênteses. Por exemplo, as expressões:**

```
a*b+c;
a*(b+c);
(a+b)*c;
(a+b)*(c+d);
```

seriam representadas utilizando notação posfixa como:

```
ab*c+
abc+*
ab+c*
ab+cd+*
```

Em ambientes Unix, essas expressões podem ser avaliadas utilizando o comando `dc` (manual: `http://compute.cnr.berkeley.edu/-cgi-bin/man-cgi?dc+1`) como a seguir:

```
$ dc <<< '2 3 5 + * p'
16
```

Escreva um programa que calcule expressões em notação posfixa. Para tal, utilize uma estrutura de dados denominada pilha como:

```
#define MAX 50

struct stack {
        int operandos[MAX];
        int tamanhoAtual;
} pilha;
```

Nesse programa, cada um dos operandos encontrados na expressão fornecida são explicitamente introduzidos e retirados do topo da pilha por invocações dos métodos *push* e *pop*, respectivamente. Esses métodos devem receber a pilha como parâmetro, empilhar (*push*) ou remover (*pop*) o operando e atualizar o `tamanhoAtual` da estrutura. Caso um operador seja encontrado um método deve ser invocado, retirando os operandos do topo do pilha e empilhando o resultado de sua aplicação. Por enquanto, implemente somente os operadores * (*mult*) e + (*add*). Por exemplo, a avaliação da expressão *a*(b+c)*, que em notação posfixa torna-se `a b c + *` será traduzida para o código como:

- **invocação do método `push` passando a pilha e `a` como parâmetros.**
- **invocação do método `push` passando a pilha e `b` como parâmetros.**
- **invocação do método `push` passando a pilha e `c` como parâmetros.**
- **invocação do método `add` que remove os dois operandos do topo da pilha, soma seus valores, e armazena o resultado de volta na pilha.**
- **invocação do método `mult` que remove os dois operandos do topo da pilha, multiplica seus valores, e armazena o resultado de volta na pilha.**
- **detalhe: a remoção dos operandos da pilha sempre deve ser realizada pelo método `pop`.**