Trabalho de Arquitetura de Sistema e Computadores I

Pedro Anjos (45558), Rafael Silva (45813)

02/05/2020

Conteúdo

1 Introdução

Pretende-se desenvolver uma calculadora que opere na notação polaca inversa (RPN - Reverse Polish No-tation). A calculadora deverá ler strings da consola, realizar as operações indicadas e mostrar o estado da memória da calculadora na forma de uma pilha.

2 Desenvolvimento do trabalho

No inicio do pragrama são definidos o topo e o tamanho da pilha, sendo o primeiro inicializado como zero. Nós escolhemos o tamanho 50 por uma questão de estética e para realizar várias operações.

Definimos também o valor lógico do topo como 0. Esse valor é necessário para quando é chamada a função bool vazia (), ela retorna verdadeiro caso a pilha esteja vazia;

Em relação à chamada a função bool cheia (), ela retorna verdadeiro caso a pilha esteja cheia;

Por fim, temos a instrução bool push () que tem como função receber o parâmetro, colocando no topo da pilha.

```
1     int main()
2 {
3     setlocale(LC_ALL, "Portuguese");
4     cabecalho();
5     printf("stack:\n");
6     printf("(empty)\n\n");
7     second_main();
8 }
```

A primeira instrução do main define para português, para que a mesma aceite acentos.
Já o cabecalho é chamada para servir de cabeçalho da calculadora, referindo a notação, os membros do grupo, o ano letivo em que foi produzido, e o comando de ajuda.
De seguida, são realizados dois printf que têm a função de demonstrar que a pilha onde são guardados os valores das operações encontra-se vazio.
Após realizar as funções, ela segue para a função second main, o esqueleto principal deste trabalho. Nós utilizamos esta função para tornar o main de mais fácil compreensão. Temos noção que podiamos ter introduzido o second main (chamada de funções para realizar as operações) dentro do main.

A função second main chama outras funções consoante o que é colocado pelo utilizador. Existem várias operações e comandos que o programa executa, cada vez que é introduzido o sinal operatório, ou outro comando.

```
printf("-> ");
        scanf("%s",s);
        if(strcmp(s,"+")==0)
        {
            soma();
```

Exemplo do que acabei de afirmei é este troço de código. Assim que são introduzidos os números, como o sinal operatório é o '+', ela vai saltar para a função soma.

```
    void soma()
{
    int num,num1;
    pop(&num);
    pop(&num1);
    push(num1+num);
```

São definidas então duas variáveis, num e num1. Logo após a sua definição, realizam os comandos pop e push. O primeiro vai retirar os números que estiverem no topo da pilha, e assim que realizar essas instruções, realiza a operação. Logo após o cálculo, o push coloca o resultado no topo da pilha.

O mesmo acontece para a subtração, multiplicação e divisão, em que o que difere é o push que possui os sinais operacionais indicados a cada operação.

Existe também a função swap, que define duas variáveis, que utiliza a função pop para retirar dois valores da pilha, trocado-os de posição.

No seguimento do programa, existem funções em que são definidas apenas uma variável, que é o caso das funções neg, dup, raiz e drop.

A primeira, retira o número que estiver no topo da pilha e repõe o mesmo sob a forma do seu simétrico; A segunda possui uma particularidade, como o objetivo é duplicar o seu valor, são realizados dois push, para que possa ser reposto na pilha sob a forma da sua duplicação. Já a função raiz, quando repõe o número que foi retirado, coloca-o sob a forma da sua raiz quadrada, a partir da operação sqrt.

As funções drop e clear já não realizam operações, são chamadas para possuirem outra utilidade. A primeira retira o número que estiver no topo da pilha, mostrando ao utilizador. O segundo retira os números da pilha enquanto o pop for verdadeiro. Assim que contrariar o seu valor lógico, a pilha fica vazia.

Caso seja necessário um auxílio, a instrução help constitui uma lista de operadores com as suas funcionalidades.

```
else
        {
            num=atoi(s);
            push(num);
        }
```

Caso não seja colocado qualquer operador, o atoi vai buscar uma string que se encontra no scanf, e a mesma é colocada no topo da pilha.

```
        if (strcmp(s,"off")!=0 && strcmp (s,    help     !=0)){

        printf("stack:\n");
         if(tamanho()== 0)
        {
            printf("(empty)\n");
        }
        }
       Count = 1;
        while(count<tamanho()+1 && strcmp(s,"off")!=0 && strcmp (s,
                help     !=0)
        {
            printf("%d\n",pilha[count]);
            count++;
        }
        printf("\n");
    }
    printf("\nBye!");
```

O primeiro if é utilizado para que caso seja escrito help ou off, não surja o stack. Caso contrário, irá sempre aparecer;

O segundo if é utilizado para que caso não exista nada na pilha, irá fazer printf empty.

Por fim, temos o while que faz printf de todos os valores que se encontram na pilha;

3 Conclusão

Conseguimos realizar o trabalho, a pilha encontra-se operacional, mas cometemos um erro no qual achamos inportante referir.

Quando o utilizador introduz uma expressão numérica, o programa executa e mostra passo a passo, ou seja, não obtemos logo o seu resultado.