Laboratório 8

Tipos Abstratos de Dados - TADs.

Oficina de Programação em C (ICP037)

Prof. Ronald Souza
IC/UFRJ

Objetivo
Praticar os conceitos de programação vistos na Aula 8.

Relembrando estruturas

Abaixo, um exemplo básico de uso de estruturas em C, **incluindo criação de tipo**:

```
#include <stdio.h>
typedef struct { //Criação do tipo “data”
     int dia;
     int mes;
     int ano;
} tData;

tData leData() { //Função de manipulação (leitura) do tipo “data”
    tData data;
    puts("Digite dia, mes e ano:");
    scanf("%d %d %d", &data.dia, &data.mes, &data.ano);
    return data;
}

int main() { //Lógica principal
    tData dt;
    dt = leData();
    printf("%d/%d/%d\n", dt.dia, dt.mes, dt.ano);
    return 0;
}
```

Atividade 1: Você implementará um TAD para representar (i) **um polinômio de grau máximo = 3** e que possui **uma única variável** e **coeficientes racionais**, e (ii) **um subconjunto de operações a ele aplicáveis**. Também implementará um programa que faça uso desse TAD, oferecendo ao usuário as operações que você disponibilizou. Para isso, siga as instruções abaixo.

Primeiro: crie 3 arquivos **dentro de um mesmo diretório**, com os seguintes nomes:

- main.c
- polinomio.c
- polinomio.h

Agora, execute os seguintes passos:

1) No arquivo **polinomio.h** digite ou copie o código abaixo (incluindo os comentários!):

```
typedef struct {
    float c3, c2, c1, c0;
} polinomio;
//Exemplo de representação de polinômio pela estrutura acima (de coeficientes):
//    Para c3 = 1, c2 = 5, c1 = 4 e c0 = -3, o polinômio correspondente será:
//    x^3 + 5x^2 + 4x - 3

typedef struct { //Raízes de um polinômio de grau 2.
      float x1, x2;
} solucao;

polinomio somarPoli(polinomio p1, polinomio p2);
polinomio derivadaPoli(polinomio p); //Ex.: derivada(x^3 + 5x^2 + 4x - 3) = 3x^2 + 10x + 4
solucao raizesPoli(polinomio p);     //Aceitar apenas polinômios de grau 2!
float valorPoli(polinomio p, int x); //Ex.: para x = 1, temos que x^3 + 5x^2 + 4x - 3 = 7
```

2) No arquivo **polinomio.c** implemente todas as funções assinadas no arquivo **polinomio.h**.

3) No arquivo main.c, implemente o programa principal, que oferecerá ao usuário as funcionalidades de calcular e exibir:

- a soma de dois polinômios – função `somarPoli`;
- a derivada de um polinômio – função `derivadaPoli`;
- a solução (isto é, as raízes) de um polinômio **de grau 2 apenas** – função `raizesPoli`;
- o valor do polinômio para um dado valor de *x* – função `valorPoli`;
- Encerrar o programa.

Não esqueça do comando **#include "polinomio.h"** no cabeçalho do arquivo **main.c**!

4) Para compilar, abra um terminal e o aponte para o mesmo diretório onde se encontram os arquivos acima. Então, digite o comando abaixo:

```
gcc main.c polinomio.c -o main.out -Wall
```

5) Rode o programa digitando "**./main.out**" (sem as aspas).

Atividade 2: Implemente o tipo **Pixel**, que possuirá tão somente 3 inteiros, **r**, **g** e **b**, correspondentes à intensidade (luminância) dos canais Vermelho (r), Verde (g) e Azul (b), **todos entre 0 e 255**. Em seguida:

a) Implemente um programa onde são lidos do teclado, na main(), os valores **r**, **g** e **b** de 2 pixels distintos. A partir daí, defenda os valores e, caso válidos, **crie os dois pixels**.

b) Crie uma função pixelSoma(), que recebe 2 pixels de entrada, soma os valores de suas cores correspondentes, e retorna um outro pixel, com as cores somadas, limitadas a 255.
Por exemplo: Para 2 pixels p1 e p2 de entrada, onde
`r1 = 0; g1 = 130; b1 = 245` e
`r2 = 80; g2 = 100; b2 = 55`,
o pixel de retorno terá os valores
`r = 80; g = 230; b = 255`.

c) Crie uma função pixelDif(), que recebe 2 pixels de entrada, subtrai do 1º os valores de cores correspondentes do 2º, e retorna um outro pixel, com as cores resultantes da diferença, limitadas a 0.
Por exemplo: Para 2 pixels p1 e p2 de entrada, onde
`r1 = 0; g1 = 130; b1 = 245` e
`r2 = 80; g2 = 100; b2 = 55`,
o pixel de retorno terá os valores
`r = 0; g = 30; b = 200`.

d) **Cor pastel:** Cores pastéis (ou "*candy colors*") possuem baixa saturação (ao contrário das cores "vivas") e alta luminância. Crie uma função que recebe **um único pixel de entrada** e retorna 1 se sua cor é pastel ou 0 caso contrário. Uma maneira simples de determinar se uma cor é pastel é verificando se todas as 3 regras abaixo se aplicam:
    1) O canal de **maior** valor (seja r, g ou b) é **exatamente igual a 255** (i.e. 100% de luminância).
    2) O canal de **menor** valor é, **no mínimo, 192** (75%) e, **no máximo, 224** (88%).
    3) O canal **intermediário** pode assumir **qualquer valor entre o menor e o maior, inclusive**.

e) Na main(), crie um menu onde seja possível ao usuário efetuar todas as operações acima, e obter os resultados impressos na tela.