Universidade Federal do Paraná
Setor de Ciências Exatas / Departamento de Informática – DInf
Algoritmos e Estruturas de Dados II – CI056 / 1º semestre de 2017
Professor: André Gregio & David Menotti

Trabalho Prático 1

Tipos Abstratos de Dados & Alocação Dinâmica

Polígonos

Avaliação: 10 pontos (10% do total da disciplina)

Data de divulgação: 24/03/2017
Data de entrega: 11/04/2016

Objetivos

Consiste em rever conceitos básicos de programação e alocação dinâmica de memória, bem como explorar os conceitos de Tipos Abstratos de Dados (TADs). Serão necessários conhecimentos em geometria analítica para realização deste trabalho prático.

Descrição

Você deverá implementar um tipo abstrato de dados `TPonto` para representar pontos no espaço $\mathbb{R}^2$. Esse TAD deverá armazenar as coordenadas x e y do ponto.

As operações que devem ser realizadas pelo seu TAD são:

1. Ler um ponto (leia as coordenadas);
2. Criar um ponto (forneça as coordenadas);

Você deverá implementar também um TAD `TPoligono` para armazenar um conjunto de pontos que descrevam o polígono (a quantidade de pontos varia de polígono para polígono). **Observe que um polígono é formado por pelo menos 3 pontos**.

Esse TAD deve implementar as seguintes operações:

1. Criar um polígono com três pontos;
2. Inserir um ponto no polígono;
3. Remover o i-ésimo ponto do polígono;
4. Calcular o perímetro do polígono;
5. Calcular a área do polígono;
6. Dados dois polígonos A e B, determinar um polígono C que seja a interseção de A e B;
7. Dados dois polígonos A e B, determinar um polígono C que seja a união de A e B.

A determinação das declarações das operações (ou assinaturas das funções) faz parte da avaliação do trabalho e é de sua inteira responsabilidade.

Implemente seus TADs em arquivos separados do programa principal (`TPonto.c`, `TPonto.h`, `TPoligono.c` e `TPoligono.h`). O uso de `makefile` é mandatório [1]. Se necessário, você pode criar outras funções auxiliares em seu TAD. Suas funções devem executar testes de consistência (critérios de parada, verificação de restrições, divisão por zero, etc.).

Uma vez que os TADs forem gerados, você deverá criar um programa principal (`main`) em um outro arquivo (`tp1.c`, por exemplo) para testar tais TADs, avaliando situações críticas e de erro.

O que deve ser entregue

- Código fonte dos programas em C (bem indentado e comentado).
  - A compilação do código não deve apresentar mensagens de atenção (*warnings*). Programas /códigos que apresentem essas mensagens terão a nota do trabalho depreciada.
- Documentação do trabalho, limitada a 12 páginas incluindo capa e sumário.

Entre outros detalhes, a documentação deve conter:

1. ***Introdução***: descrição dos problemas a serem resolvidos.
2. ***Implementação***: descrição das implementações. Deve ser detalhada a estrutura de dados utilizada (de preferência com diagramas e/ou figuras ilustrativos), o funcionamento das principais funções e procedimentos utilizados, o formato de entrada e saída de dados, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado. **Muito importante**: os códigos utilizados nas implementações devem ser inseridos na documentação.
3. ***Análise de Complexidade***: estudo da complexidade de tempo e espaço das funções implementadas e do programa como um todo (notação O).
4. ***Listagem de testes executados***: os testes executados devem ser apresentados, analisados e discutidos, quando conveniente.
5. ***Conclusão***: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.
6. ***Bibliografia***: bibliografia utilizada para o desenvolvimento do trabalho, incluindo sítio da Internet se for o caso. Uma referência bibliográfica deve ser mencionada (citada) no texto no local em que é utilizada.
7. ***Em LaTeX***: Caso o trabalho seja elaborado/escrito em LaTeX, ganha-se um **ponto extra**. Veja modelo de como fazer o trabalho em LaTeX:
   `http://www.inf.ufpr.br/{gregio,menotti}/ci056-171/tps/modelo.zip`
8. **Impressão**: Caso o trabalho não seja impresso usando modo frente-verso, perde-se **um ponto**.
9. ***Formato final***: mandatoriamente em PDF (`http://www.pdf995.com/`).

Como deve ser feita a entrega

A entrega DEVE ser feita via Moodle (`http://moodle.c3sl.ufpr.br/`) na forma de um **único** arquivo compactado (.zip, .tar.gz, .tgz, .rar, etc), contendo o código fonte, arquivos diversos e a documentação. Também deve ser entregue a documentação impressa na próxima aula teórica após a data de entrega do trabalho.

Comentários Gerais

- Comece a fazer este trabalho logo, enquanto o prazo para terminá-lo está tão longe quanto jamais poderá estar;
- Clareza, indentação e comentários no programa também serão alvos de avaliação.
- O trabalho é individual (grupo de UM aluno).
- Trabalhos plagiados (e FONTE) terão nota zero. Além disso, os infratores terão a maior nota de testes teóricos levada à zero, como forma de punição e coibição ao plágio acadêmico;
- Trabalhos entregues em atraso serão aceitos, todavia a nota zero será atribuída;
- Evite discussões inócuas com o professor em tentar postergar a data de entrega do referido trabalho.

Referências

[1] Bruce A. Maxwell. A simple makefile tutorial, 2010. `http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/` - consultado em 23/03/2017.