PROGRAMMING CHALLENGES
Resumo - Capítulo 1: Getting Started

1. **Dicas de Programação**

- Escreva comentários! Comente o que será feito no bloco (Como se estivesse especificando em portugol um algoritmo), é importante porque, se você não consegue falar facilmente o que vai fazer, provavelmente perderá tempo programando. Fica bem mais fácil de debugar depois também, se fizer comentários sobre cada módulo, pode dividi-los em blocos pra não ter que comentar linha por linha.

- Documente cada variável! Não é tempo perdido... como normalmente os programas terão várias variáveis, e o nome da variável pode acabar sendo abreviado, é bom porque economizará tempo para entender o que ela serve... É um ótimo investimento, em relação ao tempo de correção. Para programas pequenos também é aconselhável.

- Use constantes simbólicas. É uma boa prática a cada constante informada no problema, definir uma variável e colocá-la com o valor global, só por precaução, e para não ter inconsistências no meio do código.

- Coloque nomes nas variáveis com razão. Por exemplo, problemas de cartas: club, diamond, heart, spade. Atribua variáveis como C, D, H, S, e não t1, t2, t3, t4... t = tipo.

- Procure identificar padrões de código para criar uma função e diminuir a redundância dos códigos. Normalmente se repetiria um bloco do código apenas mudando parâmetros.

- Imprima na tela apenas debugs significativos... Coloque o nome das variáveis que deseja saber, e o valor na frente, normalmente separe das outras de alguma forma.

2. **Possíveis feedbacks para um problema**

- **Accepted (AC):** O programa está correto e é executado de acordo com os limites de tempo e memória.
- **Presentation Error (PE):** As saídas do programa estão corretas, mas não estão apresentadas no formato especificado. Exemplo: pode ter espaçamentos a mais, etc.

- **Wrong Answer (WA):** Para alguns casos de teste seu programa não resulta em uma resposta correta.
- **Compile Error (CE):** O código-fonte foi submetido com erro de compilação.
- **Runtime Error (RE):** Erro típico quando você define um vetor ou array com menos capacidade do que o necessário para o problema, ou quando você tenta acessar uma de memória inválida. Verifique também se não está fazendo divisão por zero.
- **Time Limit Exceeded (TLE):** A solução que você submeteu demorou mais tempo do que o permitido para rodar todos os testes dos juízes. Você deve aumentar a eficiência do seu programa.
- **Limite de Memória Excedido (MLE):** Seu programa tentou usar mais memória do que as configurações padrão do juiz.
- **Limite de saída excedido (OL):** Provavelmente seu programa está preso em um loop infinito.

3. Tipos de dados elementares

Arrays: Este tipo de dados de trabalho permite que os dados sejam acessados através da posição de cada elemento.

Arrays Multidimensionais: Eles são arrays de arrays. Um array de duas dimensões, por exemplo, pode ser imaginado como uma matriz (ou uma tabela).

Registros: São usados para agrupar registros de dados heterogêneos. Por exemplo, uma série de pessoas-registros podem agrupam os nomes das pessoas, números de identificação, alturas e pesos em um pacote simples. Abaixo temos o exemplo de um registro para representar um ponto no plano:

```
struct point
{
        int x, y;
};
```