Direcionamentos básicos para AEDSIII

Este tutorial tem como objetivo dar alguns direcionamentos para os erros comuns cometidos por pessoas iniciantes.

1 Como resolver uma PA

Esse direcionamento tem como objetivo auxiliar você a resolver uma PA. Em edições passadas, a impressão que os alunos passaram foi de resolver as PAs da seguinte forma:

1. Ler o enunciado
2. Implementar a primeira ideia que vem à cabeca
3. Descobrir que ela deu errado
4. Aplicar a ideia abaixo:

```
while (!accept_no_uri) {
  muda_alguma_coisa_que_parece_errada();
  submeter_de_novo();
}
```

O resultado disso é que você pode demorar bastante tempo para resolver as práticas. O motivo é que essa estratégia de solução lembra um pouco o teorema do macaco infinito: fazer um monte de mudanças essencialmente aleatórias e esperar que uma hora a coisa dê certo. Probabilisticamente ela vai dar. Mas nada garante que isso ocorra rápido. Resultado: você perde tempo e, quando não conseguem enviar algo que funcione, perde pontos também.

Ao invés de usarem a estratégia acima, considere fazer o seguinte:

1. Leia o enunciado várias vezes
2. Certifique-se de que você entendeu o problema direito.
   (a) O que foi pedido?
   (b) Qual o formato da entrada?
   (c) Há multiplos casos de teste? Como eles são separados um do outro?
   (d) Você consigue entender porque cada caso de teste tem a resposta dada?

(e) Consigue criar novos casos de teste do zero?

(f) Consigue criar novos casos de teste a partir dos que foram fornecidos?

3. Pense num algoritmo para resolver o problema.

   (a) Seu algoritmo é simples?

   (b) É possível simplificá-lo?

   (c) Ele funciona para os exemplos de entrada?

   (d) Consigo argumentar que ele sempre funciona?

   (e) Tem certeza que seu algoritmo é simples?

   (f) A complexidade o seu algoritmo é boa o bastante?

   (g) Tem certeza de que ela é boa o bastante?

4. Planeje uma forma de implementar seu algoritmo em C

   (a) Sua implementação é simples?

   (b) É possível torná-la mais simples?

5. Escreva seu código em C cuidadosamente

6. Leia e confira o que você escreveu

   (a) Você sabe o que cada comando no programa faz em detalhes?

   (b) A entrada e a saída parecem funcionar?

   (c) Todas as quebras de linha estão sendo tratadas corretamente?

   (d) Sempre que você acessa um array você acessa ele numa posição válida?

   (e) Se você usa ponteiros, existe alguma chance de você dereferenciar um ponteiro NULL?

   (f) Todos os arrays e strings tem tamanho adequado para armazenar as informações de que você precisa mesmo no pior caso?

   (g) Seu código parece fazer o que deveria para os exemplos de entrada?

   (h) A sintaxe parece correta?

   (i) Você zera tudo a cada caso de teste?

7. Compile o seu código no seu próprio computador

   (a) Seu código compilou com sucesso? Se não, leia o primeiro erro que o compilador imprimiu. Entenda o erro, arrume o código e volte ao passo 7.

8. Rode seu código para os exemplos do enunciado e os que você foi capaz de criar.

(a) Seu código funcionou para todos os exemplos? Vá para o passo 9.

(b) Encontrou algum problema?

- Tente resolver o exemplo do problema no papel
- Descubra o que seu código deveria fazer nesse exemplo
- Leia o código e tente descobrir se ele faz o que deveria.
- Se necessário, use printf() para descobrir qual trecho do código está se comportando de maneira diferente do esperado
- Entenda direito o porque disso estar acontecendo
- Pense cuidadosamente numa forma de arrumar o bug.
- Confira se a sua correção vai dar certo
- Implemente-a cuidadosamente.
- Confira a implementação
- Volte ao passo 7

9. Envie o código para o URI

(a) Passou? Pronto!

(b) Em caso de Time Limit Exceeded:

- Confira se seu algoritmo ou código não entra em loop infinito
- Tente gerar um exemplo muito grande e veja se seu código demora muito

(c) Em caso de wrong answer.

- Confira tudo o que você fez. Seu algoritmo está correto? Seu código está correto?
- Tente criar ainda mais exemplos que explorem coisas que você erra normalmente. Execute o passo 8 nesses exemplos

(d) Em caso de Runtime Error.

- Confira todos os acessos a arrays e usos de ponteiros
- Rode os exemplos que você fez com valgrind

(e) Em caso de presentation error

- Confira se você imprime as quebras de linhas e espaços em branco da maneira que o enunciado pede
- Confira se você imprime exatamente da forma que foi pedida

2 Tratando múltiplos casos de testes.

Muitas vezes, os problemas requerem que o programa trate múltiplos casos de testes contidos uma mesma entrada. Em geral, existem três formas comuns de se definir múltiplos casos de testes em uma mesma entrada. Nas seções seguintes, são explicadas cada forma e como você pode tratá-las.