Como Aprender a Programar

1. Introdução

Nenhuma linguagem de programação pode ser aprendida simplesmente estudando-se material escrito ou assistindo-se aulas. Isto é, um bom livro de programação ou um bom professor pode ajudar o aluno a entender as construções de uma determinada linguagem e a convencê-lo a adotar boas práticas de programação. Mas, nem um nem outro podem ser diretamente responsabilizados pela formação de um bom ou mau programador. Programação é algo que se aprende apenas com muita prática, perseverança e durante um período considerável de tempo. Infelizmente, **não existe nenhum atalho**! Portanto, seja paciente porque este processo requer muito (muito mesmo) tempo de aprendizagem e afaste-se de livros que prometem ensiná-lo a programar em 24 horas!

O presente documento apresenta algumas sugestões para um melhor aprendizado das técnicas de programação apresentadas na disciplina *Introdução à Programação*. Seguindo-as, provavelmente, elas lhe trarão bons frutos.

2. Instale seu Próprio Laboratório

É essencial para a aprendizagem de programação o uso de um laboratório consistindo de um computador desktop (ou laptop) e um ambiente de desenvolvimento (software). Praticamente, qualquer computador com menos de quinze anos de fabricação serve como hardware e o software necessário é gratuito e facilmente encontrado na internet.

Existem duas opções para obtenção de um ambiente de desenvolvimento:

- **IDE**. Tipicamente, um programa desta natureza oferece inúmeras facilidades para construção de programas que não são necessárias para um iniciante em programação. Isto é, as únicas facilidades realmente necessárias para um aprendiz são: editor de programas, compilador e linker.
- **Editor de programas avulso e compilador/linker**. Neste caso, o aprendiz usa exatamente aquilo que ele precisa, mas o uso do programa *compila* facilita as tarefas de compilação e ligação utilizando linha de comando.

As tabelas a seguir apresentam sugestões de ferramentas de desenvolvimento para as plataformas Linux e Windows.

Linux	
Software	**Comentários**
IDE: Anjuta (Gnome) ou KDevelop (KDE)	• São bons ambientes de programação, mas são um tanto complexos para aprender a programar. Portanto, evite-os se ainda não conhecê-los bem.
Compilador/linker: GCC	• Instalado automaticamente em qualquer distribuição Linux. • Apesar de gratuito, é um dos melhores compiladores de C. • Utilize a versão mais recente, pois a compatibilidade com o padrão C99 está em contínuo desenvolvimento.
Editor de programas: SciTE KWrite (KDE), vi ou Emacs	• Os editores SciTE e KWrite são bem mais fáceis de usar do que vi ou Emacs, e, portanto, são mais recomendáveis nessa disciplina. • Se você usar algum IDE, não precisará de editor de programas avulso.
compila	• É um programa que facilita a tarefa de compilação e ligação de programas escritos usando um editor de programas avulso. • Se você usar algum IDE, não precisará deste programa.
Biblioteca LeituraFacil	• Esta biblioteca foi desenvolvida com o intuito de evitar as dificuldades e frustrações com que se depara um iniciante em programação em C, tornando leitura de dados via teclado extremamente simples e robusta.

Windows	
Software	**Comentários**
IDE: CodeBlocks	• É um excelente ambiente de desenvolvimento e o iniciante não deve se assustar com ele, pois precisará usar apenas alguns poucos de seus inúmeros recursos. • É necessário instalar pelo menos um compilador e um linker para C. • Existe versão para download que inclui MinGW (v. abaixo)
MinGW	• É uma coleção de ferramentas de desenvolvimento, dentre as quais se destaca uma versão do compilador/linker GCC para Windows. • Pode ser instalado junto com o CodeBlocks ao mesmo tempo.
Editor de programas: TextPad	• Fácil e agradável de usar, mas não possui muitos recursos. • Se você decidir usar CodeBlocks, não precisará dele.
compila	• Ver comentários na tabela anterior. • Se você decidir usar CodeBlocks, não precisará dele.

No site dedicado à disciplina e na **Unidade 1** encontram-se recomendações para instalação de um ambiente de desenvolvimento adequado para acompanhamento da disciplina. **Instale seu laboratório o quanto antes e comece já a familiarizar-se com ele**!

3. Use os Exemplos de Programação Ativamente

Em vez de simplesmente ler os programas apresentados como exemplos nessa disciplina, execute-os e tente entender como eles funcionam. Arquivos contendo todos estes exemplos podem ser obtidos no site da disciplina.

Faça modificações nos referidos exemplos de programação e tente responder questões associadas aos mesmos. Por exemplo:

- O que acontece quando eu troco uma instrução por outra semelhante?
- Se eu alterar o programa, ele continuará funcionando satisfatoriamente?
- O desempenho do programa melhora com esta mudança?
- Essa mudança torna a solução mais legível ou mais fácil de modificar?
- Etc.

Quando um programa deixar de funcionar após você tê-lo alterado, tente entender o porquê. Testando suas próprias idéias a respeito de um dado programa você certamente obterá um melhor entendimento sobre como funcionam programas escritos em C.

4. Tente Resolver Todos os Exercícios de Programação

Para os exercícios de programação propostos ao final de cada unidade, siga rigorosamente o roteiro proposto para construção de algoritmos e programas apresentado nas **Unidades 2** e **3**, principalmente quando estiver lidando com problemas mais complexos.

Resista à tentação de querer resolver todos os problemas diretamente no computador. Aos poucos, você estará apto a proceder assim com a maioria dos problemas do nível daqueles propostos nessa disciplina introdutória, mas mesmo programadores experientes precisam debruçar-se sobre alguns problemas longe de um computador para planejar suas soluções.

Embora você seja encorajado a discutir ou trabalhar com outros colegas na elaboração de uma solução para um exercício de programação, sugere-se que a implementação de cada solução seja individual. Em geral, é crucial que você apresente não apenas um programa que simplesmente funcione, mas leve ainda em consideração as recomendações de estilo apresentadas neste livro e que classifiquem seu programa como um produto de qualidade. Um bom programa reflete um bom projeto de solução para o respectivo problema. Portanto, pense bastante sobre a linha de solução (algoritmo) a ser seguida antes de começar a codificar seu programa.

5. Adote um Estilo de Programação Coerente e Bem Fundamentado

Assim como não existe um estilo único de escrita de redações, também não existe um estilo único de escrita de programas. Mas, se você aderir aos conselhos básicos de estilo de programação apresentados nessa disciplina, seu estilo estará bem fundamentado.

Enquanto não adquire experiência e capacidade de discernir entre o que é certo e o que não é recomendável, evite examinar material sobre programação encontrado na internet. Democrática como é, a internet permite a qualquer um submeter qualquer coisa. Existe muito material de boa qualidade sobre programação na internet, mas, infelizmente, a maior parte do material é de má qualidade.

6. Pratique, Pratique, Pratique...

Em programação, muitos problemas são recorrentes, de modo que, com alguma experiência, um programador é capaz de identificar semelhanças entre um novo problema de programação e um problema já resolvido e incorporar partes da solução do problema conhecido na solução do novo problema. Mesmo quando dois programas parecem ser completamente diferentes, sempre há algo de um programa que se pode usar em outro por meio da simples técnica de copiar trechos de um programa e colar em outro[1]. Portanto, quando estiver resolvendo um problema de programação, tente encontrar semelhanças e diferenças entre o problema em questão e outros problemas resolvidos antes. Talvez, você possa utilizar algum programa antigo como base e modificá-lo, acrescentar ou remover partes, etc.

Uma importante diferença entre um programador experiente e um iniciante é que o primeiro reúne uma grande coletânea de problemas de programação resolvidos que podem ser recuperados de sua memória tão logo ele se depare com um novo problema que apresente similaridades com aqueles já conhecidos. Por isso, muito raramente, um programador experiente começa a escrever algum programa a partir de nada, como fazem os iniciantes. Isto é, para um programador experiente, sempre há um programa que ele já escreveu que pode servir de base para um novo programa, seja por meio de alterações, acréscimos ou subtrações de código.

Concluindo, quanto mais você praticar, mais estará apto a beneficiar-se com o conhecimento adquirido na construção de novos programas. Portanto, pratique, pratique, pratique...

---

[1] Esta técnica constitui uma forma rudimentar de **reuso de software** e será discutida em maiores detalhes na **Unidade 6**.

7. Como Programas São Avaliados

A metodologia de avaliação de programas solicitados ao aluno em provas ou como listas de exercícios valorizam critérios de **funcionalidade de programa** e **estilo de programação**. Precisamente, considerando-se que cada programa vale de 0 a 10 pontos, a avaliação em cada um dos referidos critérios corresponde a, no máximo, cinco pontos. As tabelas a seguir apresentam algumas prováveis características apresentadas por programas classificados com os respectivos pontos de acordo com estes critérios.

Funcionalidade do Programa	
Pontuação	**Características do Programa**
5	• Absolutamente (ou, pelo menos, convincentemente) correto e robusto.
4	• Quase correto, mas contém alguns *bugs* triviais que não comprometem o funcionamento do programa como um todo. • Não prevê todas as entradas possíveis para o programa (inclusive aquelas inválidas) e pode abortar em situações não usuais.
3	• Basicamente correto, mas contém alguns erros que comprometem seu funcionamento. • É abortado com relativa facilidade.
2	• Abordagem de solução do problema é razoável, mas contém muitos erros de implementação. • É abortado com muita facilidade.
1	• Basicamente, apenas sintaticamente correto. • O programa funciona, mas apresenta resultados errados.
0	• Não consegue sequer ser compilado devido a erros de sintaxe.

Estilo de Programação	
Pontuação	**Características do Programa**
5	• Utiliza algoritmos e tipos de dados adequados. • Segue todas as recomendações básicas de estilo.
4	• Utiliza algoritmos e tipos de dados adequados. • Apresenta uma interface do usuário fácil de usar. • É bem documentado. • É tão simples quanto possível. • Os identificadores são bem escolhidos. • Contém alguns poucos deslizes de estilo.
3	• Algoritmos e tipos de dados não foram bem escolhidos. • A interface do usuário é rudimentar. • Documentação e clareza são descuidadas. • É mais complexo do que deveria. • Os identificadores não são representativos, mas é razoavelmente fácil de entender.
2	• Algoritmos e tipos de dados ruins. • Apenas o próprio programador sabe usar a interface. • Documentação ruim ou ausente. • Difícil de ser entendido.
1	• Não segue um algoritmo claro ou os tipos de dados são escolhidos a esmo. • A interface do usuário é tão ruim que, às vezes, o próprio programador esquece como usá-la. • Entende-se com muito sacrifício.
0	• Ilegível. • Não é um trabalho digno de um aluno de programação.