Capítulo 1

Uma primeira suite de programas

1.1 Como rodar um programa.

Depende do que você dispõe como *ambiente* de programação.

Infelizmente alguns *ambientes* tem mais o objetivo de se *apoderarem* do usuário do que ajudá-lo a ser um *indivíduo livre e criativo*. Mas, se você comprou este livro, então você quer ser *livre e criativo*, logo se prepare para descobrir as coisas por si próprio e conte com algum auxílio por parte deste livro, mas não espere que o *livro seja uma muleta para quem não quer superar as suas próprias dificuldades.* Use o endereço eletrônico do autor[1] para tirar algumas dúvidas, mas faça isto de forma moderada. Discuta com outros colegas que já dominam um pouco assunto, este é certamente a melhor forma de evoluir em qualquer ramo do conhecimento: trabalho em equipe.

Vamos discutir alguns *ambientes de programação*, para ser franco, três ambientes: `C da Borland`, `C da Microsoft`, e o `C da Fundação GNU` dentro de um ambiente `LinuX`.

Vamos dar discutir com mais atenção o primeiro, `C da Borland`, que é considerado por todos trabalham como esta linguagem como o melhor existente para Windows. Também existe um ambiente mais antigo, ainda em franco uso, que é `Turbo C`. O que dissermos sobre o `C da Borland` vale muito aproximadamente para `Turbo C`.

O ambiente do `C da Microsoft` segue os padrões habituais de ambientes gráficos dentro do Windows, de formas que, se você estiver acostumado a trablhar dentro deste sistema, rapidamente poderá adaptar o que dissermos sobre `C da Borland` para o ambiente da Microsoft.

Observe, entretanto, que este livro não é um manual para estes ambientes, e sim um livro para ensiná-lo a programar em $\mathcal{C}$, portanto a nossas discussão

[1] tarcisio@member.ams.org

sobre *ambientes de determinados pacotes*, tem que ser breve. Além do mais, todos estes pacotes computacionais tem manuais que lhe poderão apresentar suas possiblidades de forma muito mais efetiva do que nós poderiamos fazer aqui.

A melhor forma para dominar estes *ambientes integrados* de programação consiste em gastar algum tempo descobrindo a funcionalidade dos botões que eles oferecem. Você poderá fazer isto com tranquilidade e sem o menor receio de estragar o sistema, porque ele foi feito para ser usado. A *pior coisa* que poderia acontecer seria que você *apagar algum programa gravando outro por cima*, e isto com certeza vai acontecer em algum momento, portanto é melhor que aconteça logo no começo quando o prejuizo ainda será pequeno... Portanto perca algum tempo experimentando os botões do *ambiente integrado*.

Os dois outros ambientes serão apenas citados, se você não tiver opção para trabalhar em LinuX, deverá completar o conteúdo deste livro com o manual do C correspondente, mas fique tranqüilo, as diferenças são pequenas e são importantes apenas no começo.

1. **O ambiente BC** Suponhamos que você esteja no Windows e que esteja usando o BC, Borland C. Como já disse, gaste algum tempo para reconhecer o *ambiente integrado*[2] do BC, o IDE[3]. Ele se chama assim, *ambiente integrado*, porque lhe oferece um *atelier* onde produzir os programas, guardá-los, e automaticamente rodá-los. Observo que você também corre o risco se tornar excessivamente dependente do *ambiente integrado*, procure evitar esta dependência, e faça um uso inteligente do ambiente, aos poucos você mesmo verá o que esta *advertência* significa.

   Assim que você tiver gasto uns quinze minutos experimentando o *ambiente integrado*, passe para o quarto item desta lista.

   Ao estabelecer *quinze minutos* estamos exatamente querendo lhe dizer que não procure entender tudo que se encontra à sua disposição dentro do *ambiente integrado*, e você logo vai ver que, se aprender a programar corretamente, muitas das "ferramentas"disponíveis são inúteis, e, pelo contrário, se você vier a precisar delas isto significa que estará programando mal.... e aí será preciso, de fato, usar estas ferramentas.

   Parte do que há disponível no *ambiente integrado* só lhe será útil mais a frente, quando seus programas ganharem mais densidade e já estiverem caminhando na direção de um *projeto*.

   Digamos que, no momento, o mais importante é aprender a

   - *abrir* um arquivo, (código fonte), encontrar um arquivo no disco; Para isto use o botão `File`. Experimente agora, clique no botão e vai cair um menu com
     (a) `new`, (novo) se você quiser começar um novo programa. Nunca faça isto! Entre os meus programa tem um que se chama `esqueleto.c`,

[2] muito semelhante ao ambiente do `Turbo C`
[3] *IDE - Integrated Development Environment*

comece abrindo este programa para não começar do zero... Crie um `esqueleto.c` para você.... Veja abaixo o que você pode fazer com "`esqueleto.c`" !

(b) `open` para você abrir um programa existente no disco. Você pode indicar o *caminho* onde o `BC` deve procurar o arquivo;

(c) `save` para você gravar o programa que estiver escrevendo, observe que basta acionar F2

(d) `save as`, (*gravar como*), para você escolher um outro nome de arquivo onde gravar o programa. Use esta opção com o "`esqueleto.c`". Abra `esqueleto.c` e o grave com o nome que desejar. Você já terá o novo programa na sua frente depois que fizer isto. Experimente, abra "`esqueleto.c`" e o grave como "`teste.c`".

(e) `change dir` é para mudar diretório, provavelmente pouco útil no começo.

(f) `print` para enviar para a impressora uma cópia do programa que estiver na tela.

(g) `DOS shell` para usar o `DOS`, pouco útil para os usuários do `windows`...

(h) `quit` quando você quiser ir embora...

- *procurar* uma palavra num arquivo e trocar palavras erradas e isto você vai fazer com o botão `search`, (`procura`). Nos editores de texto, em geral isto se faz com o botão `edit`, aqui não. Se você quiser traduzir para o inglês os nosso programas, vai usar este botão. Nele tem

  (a) `find` para procurar uma palavra.

  (b) `replace` para procurar e trocar palavras. Vão aparecer dois campos, no primeiro para você indicar qual a palavra que deve ser trocada, no segundo, o que a deve substituir.
  Há várias opções para você ligar ou desligar `sensível à maíuscula`, `palavras completas`, `expressões regulares`, `pergunta ao trocar`, `forward` (pra frente), `backward` (pra trás), `from cursor` (a partir do cursor), `entire scope` (no documento todo), `OK`, `change all` (muda tudo), `cancel` e `help`...

  (c) `go to line number` (vai para uma linha de número), e espera que você indique o número da linha.

- *rodar* o programa guardado num arquivo. Você vai usar o botâo `run`. Quando clicar cai um menu contendo

  (a) `run` que vai rodar o programa

  (b) `program reset`, botão importantíssimo.
  Quando você tiver rodado um programa e, depois, fizer modificações, se pedir para rodar, o `BC` vai rodar o anterior.
  Clique no `program reset` - (renova o programa), e depois no `run`.
  Este é um erro comum, se você alterar um programa e tudo voltar acontecer como antes, se lembre de fazer o `reset`.

- `compilar` um programa. Você vai fazer isto com o botão `compile`. Clique no botão e ao cair o menu, escolha `compile`.
- configurar os diretórios de trabalho Se você tiver instalado o pacote usando o instalador então não tem porque se preocupar com este item. Se sua instalação não for a padronizada você corre riscos de que o `C da Borland` não encontre os arquivos necessários. Neste caso você deve ir ao botão `Options`, lá escolher a `Directories` (`diretórios`) e registrar cuidadosamente a árvore dos diretórios onde se encontram os arquivos. Veja na figura (fig. 1.1) página 22, como se encontra

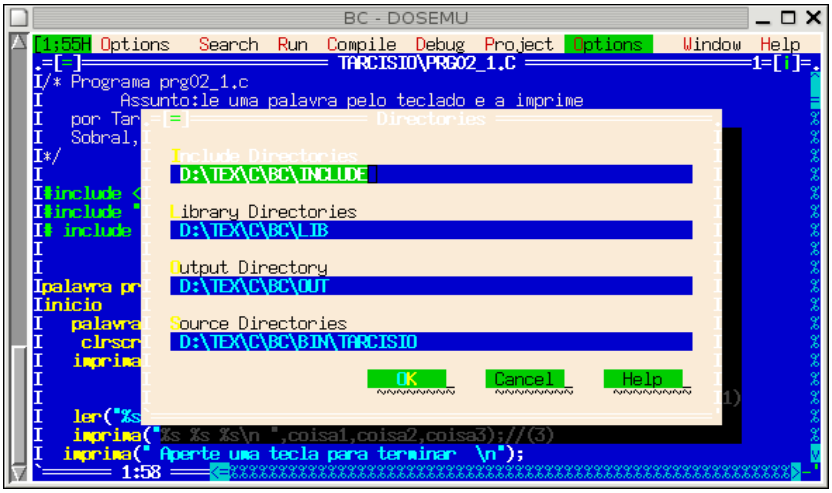

Figura 1.1: árvore de diretórios - `BC`

organizada a árvore de diretório no meu micro de trabalho. É preciso indicar o disco e todo o caminho anterior aos diretórios que `BC` procura:

```
BGI BIN OUT BINOUT INCLUDE LIB
```

Ao abrir o item `Directories` você já deve encontrar uma seção de arquivos. Veja se tudo está de acordo com sua instalação.

- `help`

  `C da Borland` tem um auxílio (help) muito bom que é preciso aprender a usar. Infelizmente não irá funcionar com os programas escritos em português. Experimente o programa[4] `primeiro01.c`. O comando

[4] os programas para `BC` ganharam nomes mais curtos, em vez de `primeiro01.c`, procure `prim01.c`

inicial do programa é `clsscr();` que serve para limpar a tela. Coloque o cursor sobre esta palavra e aperte `ctrl-F1`. O resultado é uma pequena[5] janela com informações específicas sobre este comando. Aprenda a fazer uso destas informações, elas são um manual da linguagem *on-line*. Para sair do `help`, acione a tecla `ESC`.

Se você apertar `F1` virá o manual do `C da Borland`. É um conjunto de várias janelas descrevendo toda a linguagem. Eu não poderia deixar de sugerir que você se habituasse a ler este manual *on-line*. Infelizmente em *inglês*, mas se você não se acostumar a, pelo menos, ler em *inglês*, ficará cortado de grande parte das informações tecnico-científicas. Você sai do manual acionando a tecla `ESC`.

2. **O ambiente C da Microsoft** Se você estiver o usando o `Microsoft C`, também você vai dispor de um *ambiente integrado* bem parecido com o *ambiente* do Borland C.

   Leia o item anterior e gaste *uns 15 minutos* para ganhar experiência com o `Microsoft C` e depois passe para o quarto item desta lista. O objetivo principal é carregar um programa para dentro do editor de textos e depois rodá-lo. Valem as mesmas observações que já fizemos sobre a configuração da árvore de diretórios. Nenhum sistema operacional pode *advinhar* onde se encontram os seus programas ou os arquivos de dados, tudo isto tem que ser registrado nas opções.

   Como sempre, o ideal é instalar os pacotes usando o programa apropriado para isto, ele se ocupará de toda a configuração básica. Não fazer isto é querer *dores de cabeça*. Se você for um usuário experiente, poderá, possivelmente, brincar com a configuração, caso contrário use os instaladores.

3. **Em ambiente Linux** Em geral ninguém instala `Linux` manualmente[6], tudo é feito por um instalador que vem junto com a distribuição adquirida. Estes instaladores, habitualmente, deixam a linguagem $\mathcal{C}$ instalada corretamente, até mesmo porque $\mathcal{C}$ é a linguagem natural para `Linux`, de modo que, tudo que você tem que fazer é trabalhar com seus programas no seu diretório pessoal.

   Em LinuX você conta com diversos *ambientes integrados*, por exemplo, `wpe`, `xwpe`, `xcoral`, `xemacs` ou o espartano `joe`, para citar alguns.

   Em alguns deles você deve indicar o modo com que deseja trabalhar, *o modo C* . O `xemacs` entra no *modo C* automáticamente se você abrir um arquivo com extensão ".c". Se o `xemacs` estiver bem instalado você pode contar com um ambiente integrado muito poderoso lhe oferecendo inclusive uma ajuda "on-line" sobre os conceitos da linguagem. Aprenda a usar o ambiente integrado que você tiver escolhido. Se você tiver paciência para usá-lo, terá uma poderosa ferramenta nas mãos.

[5] se não funcionar, coloque o cursor sobre a palavra, clique no botão `help` e, no menu que cair, escolha `topic search`; procure `help` no índice remissivo

[6] nem Linux, e nem nenhum outro sistema complexo

Se tiver escolhido `joe` aí você vai penar um pouquinho mais, mas foi porque você mesmo quis...eu uso o `joe` e me dou muito bem. `joe` é um editor de textos muito poderoso mas difícil de usar pois é orientado a comandos no padrão dos editores da Borland, do Turbo C ou do Turbo Pascal, ou ainda do "wordstar" dos anos 80 que foi um poderoso ambiente de edição para aquela época.

Ainda existe uma outra possibilidade, seria usar o `vi`, mas se esta for a sua escolha, então, fora de dúvidas você pertence a facção mais xiita do Unix e eu não tenho dúvidas de que você está lendo este livro somente para sacan(*) o autor... porque certamente você deve ser um exímio programador em C.

4. **Começando a trabalhar** Suponhamos que você já tenha ganho experiência com o *ambiente integrado* de sua escolha. Abra agora uma janela de edição. Escolha o primeiro botão à esquerda onde estiver escrito "`FILE`"ou "`ARQUIVO`".

   Se o ratinho[7]não estiver disponível, você pode chegar ao menu superior com `F10`[8] e depois manipular os botões do menu usando as setas para esquerda, direita ou para baixo.

   Clique com o ratinho, (ou com a seta para baixo) e escolha "`open`" ou "`abrir`", para abrir um dos programas que você já deve ter gravado no HD. Este início será semelhante em todos os *ambientes*. Escolha o primeiro programa abaixo e rode-o. Os manuais que acompanham BC ou Microsoft C trazem programas bem elementares do tipo "`primeiro.c`"e lhe informam como rodá-lo. É tudo que você precisa para começar, depois aos poucos você irá aprendendo como usar melhor o *ambiente* até o limite do necessário, (ou se preferir, fique um *expert* no uso destes ambientes...).

5. Em LinuX a coisa pode ser tão simples como escrever numa "shell"[9]

   gcc -v primeiro.c -oprimeiro

   ou

   gcc -Wall primeiro.c -oprimeiro

   em que

   - `gcc` é o nome do compilador produzido e distribuido pela Fundação GNU, "g"indica isto, "cc"signfica "compilador c".
   - `-v` é sua solicitação de "verbosidade", você deseja que o compilador lhe diga o que fizer. Evite esta opção no começo, para não se afogar nas informações que lhe serão apresentadas. Use `-Wall` em vez de `-v`.

[7]também chamado `mouse`....

[8]em alguns casos será com `F9`, `ESC` ou `TAB`, sem dúvida, é melhor garantir que o ratinho funcione...

[9]uma área de trabalho

- `-Wall` Parecido com `-v`, mas lhe apresenta apenas as reclamações mais importantes, muito bom para quem se inicia.
- `primeiro.c` é o nome do arquivo-fonte[10].
- `-o` é a opção de compilação que indica qual é o nome do arquivo que deverá ser criado. Neste exemplo escolhi `primeiro`.

Em princípio o Borland C aceita alguma coisa do tipo:

bc primeiro.c

para fazer o que descrevemos acima, produzindo um arquivo chamado `primeiro.exe`. Porém, você terá dificuldades com o caminho para que `BC` encontre o programa que você deseja compilar. A forma mais simpels de usar `Borland C` é mesmo dentro do *ambiente integrado* onde você facilmente configura os diretórios em que se encontram os seus programas.

Vamos supor o uso do *ambiente integrado*[11]. Abra um programa, `prim01.c`, por exemplo. Com o programa visível na janela de edição, clique no botão `RUN` e seu programa será *compilado* e em seguida *executado*. Os programas deste livro podem ter[12] um defeito que será preciso corrigir, para que eles funcionem bem em `C da Borland`. Vamos descrever o "problema" e a solução:

- Quando um programa termina de ser executado, dentro do *ambiente integrado*, o *ambiente* automaticamente retorna ao texto do programa (código fonte). Conseqüência, você pode não ver o resultado do programa. Para programa pequenos, como os nossos *primeiros* programas, você pode ficar com a sensação de que nada aconteceu...
- De fato isto não é um problema, você logo verá que é uma *vantagem*, porque assim o *ambiente* o trás de volta ao texto do programa (código fonte) colocando uma marca vermelha em cima de algum erro que o compilador tenha encontrado.
- Mas se não houver erros, é decepcionante... e você poderá evitar acrescentando no programa, antes do comando `return`[13], o comando `getchar()` Tentamos incluir em todos os programas o comando `pausar()` que é uma *redefinição* do `getchar()`, mas poderemos ter esquecido em algum caso. Se não acontecer nada, verifique ao final do programa se não falta `pausar()` ou `getchar()`.

Dentro do Windows/DOS, um arquivo só pode ser executado, se terminar com as extensões

.exe .com .bat

[10] arquivo fonte é o arquivo em que se encontra o programa que você escreveu

[11] IDE

[12] tentamos eliminar este problema, mas ainda pode ter ficado em algum programa, você deve, então ser alertado

[13] que por regra, todos os programas em $\mathcal{C}$ devem ter como última instrução

Em LinuX não é o nome que indica isto, mas alguns dados internos do arquivo e `gcc` se ocupa disto, "`primeiro", compilado com as intruções descritas acima, será um programa executável.`

Observação: 2 *Erros e problemas*

Problemas ao usar `BC`*. Observe que, ao enumerar* problemas*, não estamos sugerindo que um pacote é de baixa qualidade. Não existem* grandes programas*, sem erros, a não ser os programas mais mais simples. O defeito se encontra em* esconder *os erros. Isto é de fato indecente. Este livro tinha muitos erros que foram corrigidos com auxílio de leitores amigos.*

Dificilmente conseguiriamos listar todos os problemas, mas indicaremos alguns mais importantes no momento apropriado, como agora.

Quando você rodar um programa, se fizer uma alteração no mesmo, observe o item program reset *(*re-inicialização do programa*) dentro do menu* `RUN`*. Com freqüência o compilador guarda na memória a versão anterior do programa. Eis a razão deste botão. Aperte o botão antes de voltar a compilar ou rodar o programa.*

Se você ainda não apertou em todos os botões do ambiente integrado *que estiver usando, faça-o agora, para pelo menos ver o que eles contém.*

Se você não alterar nada, não irá estragar nada, também...olhar não faz mal[14]*.*

Vocabulário: 1 *compilar,* `compile`*, executável, rodar,* `run`

- *compilar em inglês,* `compile`*, é uma das funções do "compilador", fazer uma análise sintática do código fonte para verificar se as regras (sintaxe) da linguagem de programação foram todas respeitadas. Se isto for verdade, o compilador cria um executável.*

- *executável é um arquivo que o sistema operacional considera que pode* fazer *algum processo, produzir um resultado. Este é o objetivo principal dos programas de computador...este conceito se opõe ao de código fonte. Ao criar um executável, este programa poderá ser executado em outro computador, que roda o mesmo sistema operacional. Se você tiver compilado com o* `BC`*, poderá rodar o programa em qualquer computador sob Windows. Se você tiver compilado com* `gcc`*, poderá rodá-lo em qualquer computador sob* `Linux`[15]*.*

- *código fonte é o texto que você escreveu como programa e gravou em um arquivo no disco. Ele precisa ser compilado para que seja gerado um executável que o sistema operacional irá permitir que rode.*

- *rodar, em inglês* `run`*, executar um programa.*

Agora vamos apresentar-lhe um bloco de exercícios. Você não conseguirá aprender nada sem fazer exercícios, muito menos poderá aprender a programar

[14] algumas vezes, talvez...

[15] existe um compilador gratuito, `djdev` para `DOS/Windows`, veja no índice remissivo

sem fazer *exercícios de programação*[16]. A grande maioria dos programas deste livro, são exercícios. Os programas "dialogam" com você pedindo que você volte a ler o programa e corrija erros que deixamos nos programas. Em geral o programa seguinte contém a correção de um erro, e mais outro erro...

Exercícios: 1 *Os primeiros programas*

O objetivo deste bloco de exercícios é a compreensão do programa `primeiro.c`

Vamos então pedir que você rode e leia, nesta ordem, os programas `primeiro01.c`[a]`, primeiro02.c, ... primeiro07.c` *que irão desembocar em* `primeiro.c` *que se encontra editado a seguir.*

[a]para DOS,Windows, use os programas com nomes curtos, prim01.c, prim02.c,... porque primeiro01.c se confunde com primeiro02.c

1. *Rode e depois leia* `primeiro01.c`.

   gcc primeiro01.c -Wall -oprog

   ./prog[17]

2. *Volte a a rodar e ler* `primeiro01.c` *procurando entender cada linha de comando do programa. Em particular leia os "comentários",* o texto inicial do programa*, e veja como ele se encontra destacado do* corpo *do programa. Analise o objetivo dos comentários, inclusive o registro de que o programa contém erros.*

3. *Corrija o comentário de* `primeiro01.c` *indicando qual é o erro o programa comete: a falta de* $\backslash n$.

4. *Rode e depois leia* `primeiro02.c`.

   gcc primeiro02.c -Wall -oprog

   ./prog

   *Altere o programa, como ele mesmo sugere, e tente compilá-lo, analise a mensagem de erro. Experimente apagar alguns "ponto-e-vírgulas", compile e analise a mensagem de erro. A falta de um único "*ponto e vírgula*" pode gerar uma imensidão de mensagens de erro. Experimente, apague*[18] *um "ponto-e-virgula" e compile o programa. Esta é uma dificuldade que os compiladores têm, antes se perder na leitura de uma enxurrada de* mensagens de erro *verifique se não falta um simples "*ponto e vírgula*".*

5. *Rode o programa* `primeiro03.c`. *Ele mente, corrija-o.*

[16]repetiremos mais ainda algumas vezes esta observação, para conscientizá-lo de que não será apenas lendo, que aprenderá a programar

[17]se o sistema estiver bem instalado não será necessário "./", bastará "prog"...

[18]você não precisa apagar, basta colocar // na frente

6. *Rode os programas* `primeiro04.c, primeiro05.c` *e faça o que eles sugerem.*

7. *Leia e rode o programa* `primeiro06.c`. *Este programa é um contra-exemplo. Ficou muito grande, saiu da tela. Veja como ele é extranho, tem uma linha de número 12 que aparece várias vezes, isto é um sintoma de imperfeição...*

8. *Melhore a redação no programa* `primeiro07.c`, *tem letra maíscula depois de vírgula, e precisa de algumas mundaças de linhas... mas tome cuidado para que o resultado não fique ilegível...*

9. *Um exercício importante: como re-utilizar um programa. Escolha algum dos programas que você acabou de usar, algum que lhe tiver parecido especial. Grave-o com outro nome, por exemplo* `teste.c`[19]. *Em* `Linux` *será o mesmo se você estiver usando alguma* `IDE`, *ou numa,*[20] shell, *digite*

   ```
   cp primeiro01.c teste.c
   ```

   Agora você tem o mesmo texto de `primeiro01.c` *(ou* `prim01.c`*) dentro do arquivo (ainda não gravado)* `teste.c`. *Rode este novo programa depois que você nele faça algumas modificações que lhe pareçam interessantes. Este é o método que nós, programadores, usamos para construir novos programas...nunca começamos do zero.*

No último exercício lhe contamos o segredo de como escrever novos programas. Deixe-me contar-lhe outro. Entre os programas que você recebeu estão dois iguais, `esqueleto.c` e `padrao.c`. São este programas que usamos para começar a escrever outro.Crie os seus, com os seus dados, e com as estruturas de programação que considerar básicas.

O texto abaixo é uma cópia do arquivo `primeiro.c` que você tem no disco. Nele não há os sinais de acentuação da lingua portuguesa, é um programa de computador e não um texto em nossa lingua.

```
/* Programa primeiro.c
Assunto: Escreve algumas frases e recebe uma informacao pelo teclado.
Programa com erros na saida de dados, nao imprime o que se espera.
COMENTARIO
Este texto inicial do programa eh um COMENTARIO e
se encontra demarcado com "barra+asterisco"no
inicio e depois "asterisco+barra"ao final.
Programa define a variavel 'coisa' como um vetor de caracteres
com 30 coordenadas.
por Tarcisio Praciano Pereira - 10 licoes para aprender C
Sobral, julho de 2001 - UVA
*/
// isto aqui tambem eh um COMENTARIO
```

[19] no `BC` escolha "`save-as`" e use a caixa de diálogo que aparece para, nela, escrever `teste.c`
[20] área de trabalho

```
#include < stdio.h >
#include "traducao.h"
principal()
inicio
        palavra coisa[30]; // variavel
        imprima("%s\n", "Escreva uma frase no teclado, ");
        imprima("%s\n", "pode ser o seu nome, por exemplo: ");
        ler("%s", coisa);// leitura de dados pelo teclado
        imprima("%s\n",-----------------------------");
        imprima("Voce escreveu: %s %c\n", coisa,'?');
        imprima("%s\n",-----------------------------");
        imprima("%s\n","Agora leia o programa para ");
        imprima("%s\n","acompanhar a critica que vai ser feita.");
        imprima("%s\n",-----------------------------");
        imprima("%s\n","Observe que 'ler' nao obedeceu a regra");
        imprima("%s\n","de uso do direcionador & de enderecos,");
        imprima("%s\n","como anunciamos no texto....");
        voltar 0; // todo programa deve terminar com este comando
fim
```

Exercícios: 2 *Análise de* `primeiro.c`

1. *Rode o programa* `primeiro.c` *e depois o leia.*
2. *Ao compilar* `primeiro.c`*, você recebeu uma advertência:*

   ```
   primeiro.c:22: warning: return-type defaults to int
   ```

   porque foi omitido o tipo *de* `principal()` *e ao final o valor devolvido é zero. Um conflito. Corrija isto, dado um* tipo *para* `principal()`*:*

   ```
   inteira principal()
   ```

   A advertência se compõe de três partes, separadas por "dois pontos"que é o "separador oficial do `Unix`*:*

   - `primeiro.c` *o nome do programa;*
   - 22 *a linha em que o erro foi detectado;*
   - `warning`*, um aviso. Se o erro for mais grave será* `error`*. É o tipo de dados da função* `principal()` *que não foi fornecido.*

   Em C todas as funções ou variáveis tem que ter um tipo*.*
3. *Rode o programa* `primeiro.c` *digitando o seu nome sem espaços.*
4. *Digite uma seqüência de mais de 30 caracteres como resposta ao programa, e analise o resultado.*

A solução para que o compilador deixe de reclamar contra `primeiro.c` é definir

```
inteira principal()
```

Deixamos este erro ficar de propósito, para justificar esta observação. Ao mesmo tempo acrescentamos: somente na segunda parte é que discutiremos a fundo a questão do *tipo de dados* a que se encontra afeto este problema.

Infelizmente vamos ter que trabalhar com `tipos de dados` antes de conseguirmos explicar tudo direitinho eis a razão do erro ter ficado. Afinal, erros são parte do aprendizado...

Se você quiser avançar este assunto, veja o capítulo 7 onde discutimos *tipos de dados*, você deve, agora, fazer uma leitura rápida daquele capítulo para se inteirar desta noção.... e voltar logo para cá.

Você está aprendendo a programar em C, com os comandos traduzidos para o Português, e os programas em *português* estarão rodando...

No próximo capítulo trabalharemos com a suite de programas "prog*.c"... lá discutiremos esta notação extranha, *formatadores de dados* que aparecem dentro da função `imprima()`. Mas pode fazer uma leitura rápida do capítulo 2 agora.

Exercícios: 3 *Alterando* `primeiro.c`

Os dois últimos exercícios desta lista estão fora do contexto, e portanto são difíceis para o iniciante. Eles se eoncontram aqui apenas para mostrar alguma coisa do que pode ser feito com $\mathcal{C}$.

1. *Altere o texto dentro da função* `imprima()`*, por exemplo, mande escrever o seu nome.*

2. *Altere substancialmente o texto de todas as funções* `imprima()` *no programa, por exemplo, mande escrever os nomes dos seus amigos e amigas.*

3. *Altere o texto das funções* `imprima` *do programa, por exemplo, para escrever uma pequena lista telefônica. Pode ser a sua lista telefônica particular...*

4. *Você viu que um programa pode colocar um texto na tela, altere* `primeiro.c` *para colocar um texto seu na tela do computador, por exemplo, um lembrete sobre as coisas que você deve fazer no dia. O executável assim criado, pode ser incluido no* `autoexec.bat` *do DOS/Windows e rodar sempre que você ligar o computador.*

5. *** fora do contexto Rode o programa* `agender01_p.c` *e depois o leia:*

```
gcc -Wall -oprog agender01_p.c
./prog
```

   A primeira linha serve para compilar o programa, quer dizer, pedir ao `gcc` *que crie um programa executável a partir do código fonte. A segunda linha é para executar o arquivo executável* `prog`*.*

6. *** fora do contexto Se você quiser compor sua lista telefônica em disco, veja* `agender01_p.c`*. Tente modificar o programa, sem se preocupar com entendê-lo, e faça sua agenda. Se precupe apenas com alterar as mensagens dentro das funções* `imprima()`, `imprime_arq()`*. O arquivo produzido pelo programa poderá ser lido, e editado, depois, com qualquer editor de textos e enviado para a impressora.*