

# Laboratório de programação

## Introdução

Universidade Estadual Vale do Acaraú – UVA

---

Paulo Regis Menezes Sousa

paulo\_regis@uvanet.br

O programa mínimo em C

Criação, compilação e execução de um programa

A estrutura de um programa em C

Variáveis

Entrada e saída de dados

Operações aritméticas

- O programa mais simples que se pode escrever em C:

```
1  int main() {}
```

- Todo o programa em C deverá conter apenas uma função `main()`.
- As chaves `{` e `}` agrupam instruções em um bloco de código.
- Um comentário é qualquer texto precedido por `//` ou delimitado por `/*` e `*/`. Por exemplo:

```
1  // Comentário de uma linha só
2  int main() {
3      /* Comentário de mais
4         de uma linha */
5  }
```

- O arquivo-fonte de um código em linguagem C é salvo em um arquivo com a extensão “.c”.
- Para que o processador possa executar nosso programa, este deve ser traduzido para a linguagem de máquina.
- Essa tradução se chama **compilação** e é feita pelo programa denominado compilador.

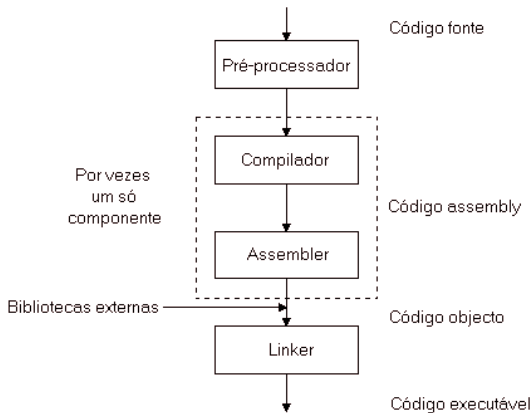
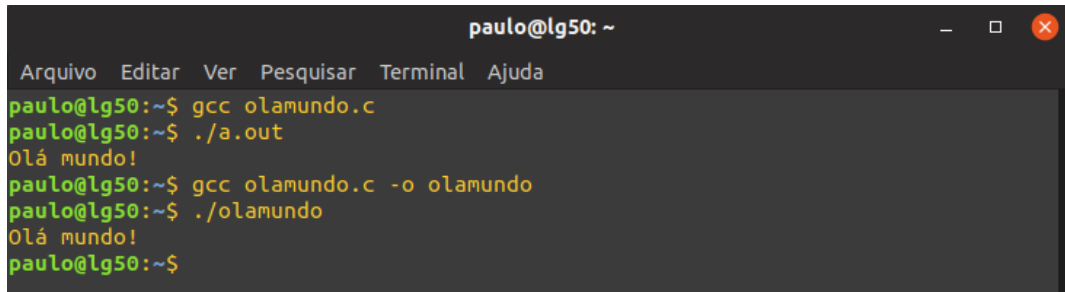


Figura: Etapas do processo de compilação.

- O **pré-processador** O pré-processador atua apenas ao nível do código fonte, modificando-o. Trabalha apenas com texto. Algumas das suas funções são:
  1. remover os comentários de um programa;
  2. interpretar diretivas especiais a si dirigidas, que começam pelo carácter #.
- O **compilador** Alguns compiladores traduzem o código fonte (texto) recebido do pré-processador para linguagem *assembly* (também texto). No entanto também é comum os compiladores serem capazes de gerarem diretamente **código objeto** (instruções do processador já em código binário).

- **assembler** O assembler traduz código em linguagem assembly (texto) para código objeto. Pode estar integrado no compilador.  
O código objeto é geralmente armazenado em arquivos com a extensão .o (unix) ou .obj (ms-dos).
- **linker** Se o programa referencia funções da biblioteca standard ou outras funções contidas em arquivos com códigos fonte diferentes do principal (que contém a função `main()`), o linker combina todos os objetos com o resultado compilado dessas funções num único arquivo com código executável.

```
1 #include <stdio.h> //Bibliotecas necessárias.
2
3 //Função principal
4 int main() {
5     printf("Olá mundo!\n"); //Imprime texto no terminal.
6     return 0;               //Indicacao de término do programa.
7 }
```



A terminal window titled "paulo@lg50: ~" with standard window controls. The menu bar includes "Arquivo", "Editar", "Ver", "Pesquisar", "Terminal", and "Ajuda". The terminal shows the following commands and output:

```
paulo@lg50:~$ gcc olamundo.c
paulo@lg50:~$ ./a.out
Olá mundo!
paulo@lg50:~$ gcc olamundo.c -o olamundo
paulo@lg50:~$ ./olamundo
Olá mundo!
paulo@lg50:~$
```

Figura: Compilação usando o GCC.

- A estrutura genérica de um programa em C é a que se apresenta a seguir, podendo alguns dos elementos não existir:
  - Comandos do pré-processador
  - Definições de tipos
  - Protótipos de funções - declaração dos tipos de retorno e dos tipos dos parâmetros das funções
  - Variáveis globais
  - Funções
- Deverá existir sempre uma função `main()`.
- As funções têm a seguinte estrutura:

```
tipo nome_da_funcao(tipo par_1, tipo par_2, ...) {  
    ... instruções em C...  
}
```



- O C tem pré-definidos os seguintes tipos de dados simples:

Tipo	Faixa de valores	Tamanho (aproximado)
<code>char</code>	-128 a 127	8 bits
<code>unsigned char</code>	0 a 255	8 bits
<code>int</code>	-32.768 a 32.767	16 bits
<code>unsigned int</code>	0 a 65.535	16 bits
<code>short int</code>	-32.768 a 32.767	16 bits
<code>long</code>	-2.147.483.648 a 2.147.483.647	32 bits
<code>unsigned long</code>	0 a 4.294.967.295	32 bits
<code>float</code>	$3.4 \times 10^{-38}$ a $3.4 \times 10^{38}$	32 bits
<code>double</code>	$1.7 \times 10^{-308}$ a $1.7 \times 10^{308}$	64 bits
<code>long double</code>	$3.4 \times 10^{-4932}$ a $1.1 \times 10^{4932}$	80 bits

Tabela: Tipos de dados em C.

- Geralmente nos sistemas UNIX os tipos **int** e **long int** são equivalentes (inteiros de 32 bits).
- No entanto noutros sistemas é possível que o tipo **int** seja equivalente a um **short int** (inteiro de 16 bits).
- O C não tem um tipo **booleano** pré-definido, no entanto, poderá usar-se um **char** (ou melhor um **unsigned char**) ou um **int** para o efeito.

- As *variáveis* são declaradas após a especificação de seu tipo.

```
1 int idade;  
2 char letra;  
3 float nota, media;
```

- É também possível inicializar as variáveis no momento da declaração. Usa-se para isso o operador de atribuição `=`.

```
1 float sum = 0.0;  
2 int bigsum = 0;  
3 char ch = 'A';
```

- As variáveis podem ser classificadas com relação ao seu escopo como **locais** e **globais**.
- Variáveis **globais** são declaradas fora de blocos de funções e são visíveis por todas as funções no programa.

```
1  int number;  
2  
3  int funcA() {  
4      number = 5; OK  
5  }  
6  
7  void funcB() {  
8      number = 77; OK  
9  }
```

- Variáveis **locais** são declaradas dentro de blocos de funções e são visíveis apenas dentro do seu bloco de código.

```
1      int funcA() {  
2          int number;  
3  
4          number = 5; OK  
5      }  
6  
7      void funcB() {  
8          number = 77; ERRO  
9      }
```

- As funções da biblioteca padrão do C `printf()` e `scanf()` permitem escrever no *console* e ler do *teclado*, respectivamente, o valor de variáveis.
- Estas funções têm como primeiro parâmetro uma **string** especificando o formato e a ordem das variáveis a escrever ou a ler.
- Seguem-se como parâmetros as próprias variáveis pela ordem especificada.
- Na string de formatação indica-se o local e o tipo de um valor de variável através do carácter % seguido de uma letra indicadora do tipo. Alguns dos tipos suportados são:
  - %c - **char**
  - %d - **int**
  - %f - **float** e **double**
  - %s - strings

- Um exemplo:

```
1 printf("Os valores de A, B e C são: %c, %d, %f\n", A, B, C);
```

- Para modificar o número de casas decimais use um ponto seguido de um número (.x) entre o % e o formatador do tipo.

```
1 #include <stdio.h>
2
3 int main() {
4     double item = 10.12304;
5     printf("%f\n", item);
6     printf("%5.2f\n", item);
7     return 0;
8 }
```

- Para a entrada de dados usamos a função `scanf()`, para que a função possa modificar os valores das variáveis passadas como parâmetro usamos o operador `&`.

```
1  #include <stdio.h>
2
3  int main() {
4      int num = 0;
5      scanf("Numero: %d\n", &num);
6      printf("Quadrado: ", num * num);
7      return 0;
8  }
```



- O C suporta as quatro operações aritméticas padrão nas suas expressões que são representadas pelos símbolos habituais (+, -, \*, /).
- Outros operadores aritméticos são os operadores de incremento e decremento, representados respectivamente por ++ e --.
- Os operadores ++ e -- podem ser pósfixos

```
1  #include <stdio.h>
2
3  int main() {
4      int x, y = 2, z = 2;
5      x = ++y + z--;
6      printf("%d\n", x);
7      return 0;
8  }
```

- Um outro operador aritmético do C é o operador módulo %. Este operador é equivalente ao operador **mod** do Pascal e tem como resultado o resto da divisão inteira.
- O operador de divisão /, pode ser usado com inteiros e reais. No entanto, se ambos os operandos forem inteiros o resultado é a divisão inteira.

1. No Code::Blocks, selecione o menu **File** → **New** → **Project ...**
2. Na caixa de diálogo **New from template** selecione **Console** na caixa de combinação **Category**.
3. Selecione o ícone **Console application** e clique no botão **Go**.
4. Na caixa de diálogo **Console application** selecione **C** na caixa de listagem **Please make a selection** e clique no botão **Next**.
5. Na caixa de diálogo **Console application**, digite **Projeto01** (ou outro nome para seu projeto, o qual não deve conter espaços ou caracteres especiais) na caixa de texto **Project title**:
6. Clique no botão ... à esquerda da caixa de texto **Folder to create project in**: e escolha uma pasta onde você possa criar novos arquivos.
7. Clique no botão **Next** e em seguida no botão **Finish** e seu projeto será criado na pasta especificada com o nome **Projeto01** (ou com o nome que você escolheu).
8. Caso seu projeto não esteja visível no lado direito da tela, clique no menu **View** → **Manager**, clique em **Sources** e dê um clique duplo sobre **main.c**. Esse é o programa inicial que você irá modificar para atender às suas necessidades.
9. Caso a guia **Logs & others** não esteja visível na parte inferior da tela, clique em **View** → **Logs**.
10. Compile o programa padrão criado no projeto do item 7 clicando no botão **Build** da barra de ferramentas, ou use o menu **Build** → **Build**, ou ainda pressione **Shift+F9**. Em seguida, execute o programa clicando no botão **Run** da barra de ferramentas, ou use o menu **Build** → **Run**, ou ainda pressione **Ctrl+F10**. A compilação e execução do programa podem também ser feitas em um único passo clicando no botão **Build and run** da barra de ferramentas, ou usando o menu **Build** → **Build and run**, ou ainda pressione **F9**. Após executar o programa, a janela do programa **Projeto01** deve surgir na tela apresentando a mensagem **Hello World !!!**. Pressione a tecla **Enter** para fechá-la e voltar para o IDE.

### Exercício 1

Leia uma velocidade em km/h (quilômetros por hora) e apresente convertida em m/s (metros por segundo). A fórmula de conversão é  $M = K/3,6$ , sendo  $K$  a velocidade em km/h e  $M$  em m/s.

### Exercício 2

Faça um programa que leia um valor em reais e a cotação do dólar. Em seguida, imprima o valor correspondente em dólares.

### Exercício 3

Leia um valor que represente uma temperatura em graus Celsius e apresente-a convertida em graus Fahrenheit. A fórmula de conversão é:  $F = C * 1.8 + 32$ , sendo  $F$  a temperatura em Fahrenheit e  $C$  a temperatura em Celsius.

## Exercício 4

Leia um ângulo em graus e apresente-o convertido em radianos. A fórmula de conversão é  $R = G * \pi / 180$ , sendo  $G$  o ângulo em graus e  $R$  em radianos e  $\pi = 3.141592$ .

## Exercício 5

Faça um programa para ler um número inteiro positivo de três dígitos. Em seguida, calcule e mostre o número formado pelos dígitos invertidos do número lido. Exemplo:  
Número lido = 123 Número gerado = 321