

Laboratório de programação

Introdução

Universidade Estadual Vale do Acaraú – UVA

Paulo Regis Menezes Sousa

paulo_regis@uvanet.br

A linguagem C

Criação, compilação e execução de um programa

A estrutura de um programa em C

Variáveis

Operações aritméticas

- A linguagem C foi inventada e implementada originalmente por Dennis Ritchie na década de 1970 e oficialmente regulamentado pelo ANSI (*American National Standards Institute*) em 1983.
- C é uma linguagem **estruturada** de **médio nível**.
- C é uma linguagem **compilada**.

Lista de palavras-chave da linguagem C

auto	double	int	struct	break	else	long	switch
case	enum	if	typeof	continue	float	return	while
union	const	for	short	unsigned	char	extern	signed
void	default	do	sizeof	volatile	goto	register	static

- Todo programa escrito na linguagem em C obedece a um esqueleto padrão:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(){
5
6      printf("Hello World\n");
7
8      return 0;
9  }
```

- Todo o programa em C deverá conter apenas uma função `main()`.
- As chaves `{` e `}` agrupam instruções delimitando um **bloco de código**.

- Podem ser inseridos comentários de dois tipos
 - Comentário de uma linha: qualquer texto precedido por //
 - Comentário de mais de uma linha: qualquer texto delimitado por /* e */

Por exemplo:

```
1 // Comentário de uma linha só
2 int main() {
3     /* Comentário de mais
4     de uma linha */
5
6     return 0;
7 }
```

- 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.

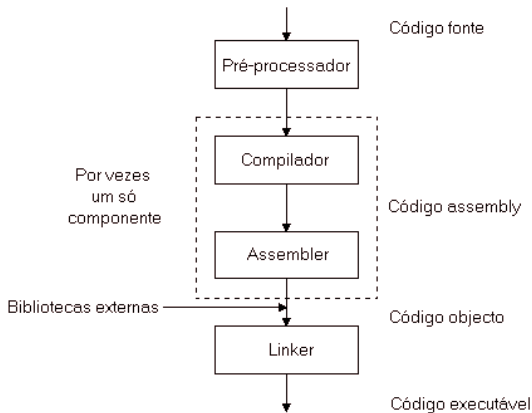
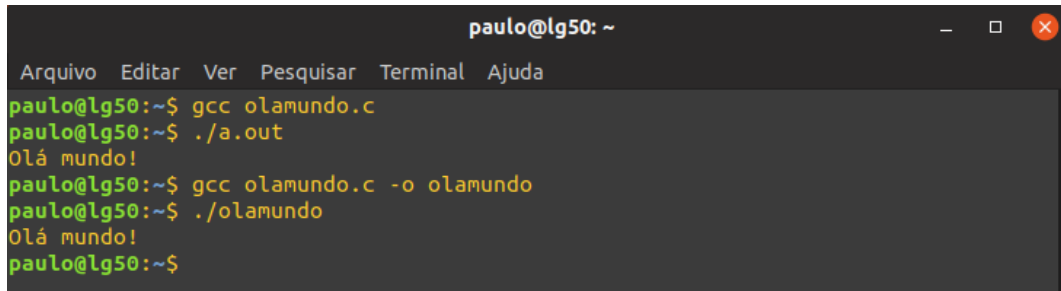


Figura: Etapas do processo de compilação.

- 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;               //Indicação 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 sequence of 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.

- 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>	3.4×10^{-38} a 3.4×10^{38}	32 bits
<code>double</code>	1.7×10^{-308} a 1.7×10^{308}	64 bits
<code>long double</code>	3.4×10^{-4932} a 1.1×10^{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 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.

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