Marisangila Alves, MSc
marisangila.alves@udesc.com
marisangila.com.br

UDESC
UNIVERSIDADE DO ESTADO DE SANTA CATARINA
JOINVILLE
CENTRO DE CIÊNCIAS TECNOLÓGICAS

UDESC
Universidade do Estado de Santa Catarina

2025/1

Algoritmo e Linguagem de Programação

Linguagem de Programação C
Fundamentos

Sumário

História

- **Origem do C:**
  - Desenvolvido a partir das linguagens BCPL e B.
  - Desenvolvido por Dennis Ritchie no Bell Laboratories em 1972.
  - Tornou-se conhecido como a linguagem de desenvolvimento do UNIX.
- **Problemas de compatibilidade e padronização:**
  - Expansão do C levou a variantes incompatíveis entre plataformas.
  - Comitê técnico X3J11 foi criado em 1983 para padronizar a linguagem.
  - Em 1989, o padrão foi aprovado. O documento é conhecido como ANSI/ISO 9899:1990.

Ferramentas de Programação

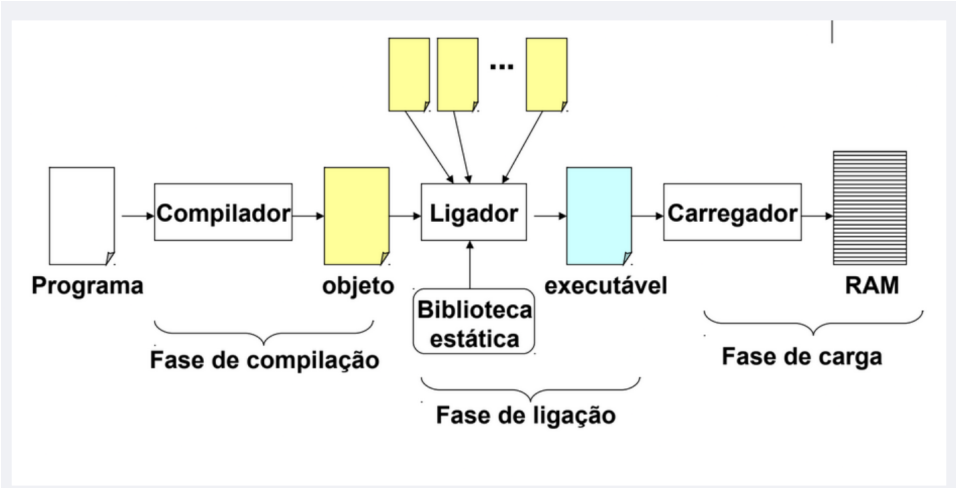

Figura 1: Processo de compilação.

1. **Compilação:**
   - O compilador traduz o código-fonte em código de máquina.
   - O arquivo de objeto (`.o`) é gerado, mas não é executável.
2. **Ligação (Linking):**
   - O linker combina arquivos de objetos e bibliotecas.
   - O arquivo executável é gerado (`.exe` ou similar).
3. **Geração do Arquivo Executável:**
   - O arquivo executável contém o código de máquina e as bibliotecas necessárias.
   - Está pronto para ser executado pelo sistema operacional.
4. **Carregamento na RAM:**
   - O carregador coloca o programa na memória RAM.
   - O programa é iniciado e executado.

(a) CLion (JetBrains, 2025)

(b) Code Blocks (Code::Blocks, 2025)

(c) DEVC++ (Bloodshed, 2025)

(d) Visual Studio Code (Microsoft, 2024)

Figura 2: Editores de texto e IDE's para C.

- Crie uma conta no Replit:
- 🔗 Acesse: Replit
- É possível compartilhar a ediçao do código.
- É possível armazenar em *cloud*.

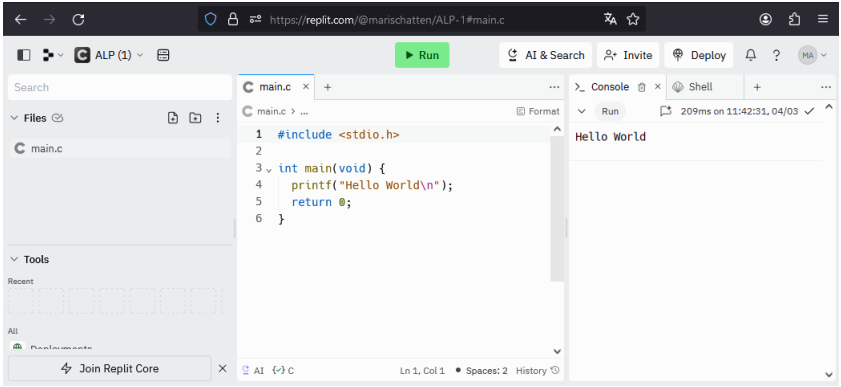

Figura 3: Ambiente de desenvolvimento *cloud*.

- É possível compilar e executar seu programa em C através da linha de comando, alternativa a interface gráfica.

Compilar e executar:

1. Para compilar: `gcc nome_arquivo.c -o nome_arquivo`
2. Para executar: `./nome_arquivo`

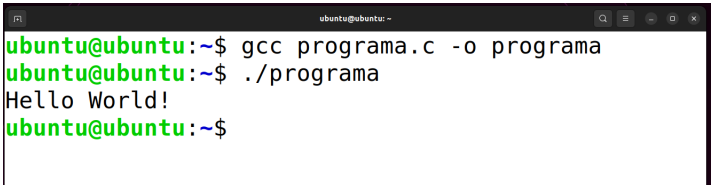

Figura 4: Compilar e executar programa em C.

- O compilador pode gerar alguns "warnings"(alertas). Para ocultar esses alertas, use o parâmetro `-w`.

Compilar e executar (ocultar alertas):

1. Para compilar: `gcc nome_arquivo.c -o nome_arquivo -w`
2. Para executar: `./nome_arquivo`

- Problemas de "linker"podem ocorrer, como o linkador não encontrar algumas bibliotecas [1] padrão, por exemplo, `math.h`. Para resolver isso, inclua o parâmetro `-lm`.

Compilar e executar (com bibliotecas de matemática):

1. Para compilar: `gcc nome_arquivo.c -o nome_arquivo -lm`
2. Para executar: `./nome_arquivo`

[1] Uma biblioteca em programação é um conjunto de códigos prontos que podem ser reutilizados em diferentes programas, facilitando a implementação de tarefas comuns.

NÃO ESQUEÇA!

Nota:

1. Para compilar: `gcc nome_arquivo.c -o nome_arquivo -w -lm`
2. Para executar: `./nome_arquivo`

Primeiro Programa

Estrutura básica.

```
1 int main(){
2     //código fonte aqui!
3     return 0;
4 }
```

Código 1: Exemplo de Hello World.

Primeiro Programa

```c
#include <stdio.h>
int main(){
    printf("Hello world!");
    return 0;
}
```

Código 2: Exemplo de Hello World.

Quebra de Linha

Para inserir uma quebra de linha, use o caractere de escape \n.

```c
#include <stdio.h>
int main(){
    printf("Hello world!\n");
    printf("Hello world!\n");
    return 0;
}
```

Código 3: Exemplo com quebra de linha.

Variáveis

- `int`: Tipo utilizado para representar números inteiros.

Exemplo de valores inteiros

`-15, -2, 5, 150, 1590`

- `float, double`: Tipos utilizados para representar números reais (com ponto decimal).

Exemplo de valores reais

`-3.5, 5.7, 8.0, 10.2, 75.2`

- `char`: Tipo utilizado para representar um único caractere.

Exemplo de valores de caractere

`"x", "X", "@", "(espaço em branco)", "7", -"`

```c
int main(){
    int idade;
    float altura;
    char inicial_nome;
    return 0;
}
```

Código 4: Exemplo de declaração de variáveis.

- Um identificador de variável é uma série de caracteres que consistem em letras, dígitos e sublinhados (_).
- Um identificador não pode começar com um dígito.
- Um identificador pode ter qualquer comprimento, mas somente os 31 primeiros caracteres serão reconhecidos pelos compiladores C, de acordo com o padrão ANSI C.
- O C faz distinção entre letras maiúsculas e minúsculas (sensível a caixa alta/baixa ou case sensitive), ou seja, as letras maiúsculas e minúsculas são diferentes em C.
- Um identificador não deve ser uma palavra reservada (Tabela 5 do apêndice).

```c
int main(){
    int quantidade_bananas;
    int quantidade_laranjas;
    int quantidade_macas;
    return 0;
}
```

Código 5: Exemplo sem agrupamento de variáveis.

```c
int main(){
    int quantidade_bananas, quantidade_laranjas, quantidade_macas;
    return 0;
}
```

Código 6: Exemplo com agrupamento de variáveis.

```c
int main(){
    int idade = 20;
    float altura = 1.65;
    char inicial_nome = 'A';
    return 0;
}
```

Código 7: Exemplo de definição de variáveis.

```c
int main(){
    int idade;
    float altura;
    char inicial_nome;

    idade = 20;
    altura = 1.65;
    inicial_nome = 'A';
    return 0;
}
```

Código 8: Exemplo de atribuição de variáveis.

Entrada

```c
#include <stdio.h>
int main(){
    int idade;
    float altura;
    char inicial_nome;

    scanf("%d" , &idade);
    scanf("%f", &altura);
    scanf(" %c", &inicial_nome);
    return 0;
}
```

Código 9: Exemplo de entrada padrão.

Formato	Sintaxe	Tipo de Dado
%d	int	inteiro
%f	float	real
%c	char [2]	caracter

Tabela 1: Formatação de Entrada.

[2] O especificador %c no scanf lê um único caractere, incluindo espaços. Sem um espaço antes, ele pode ler um caractere de espaço ou nova linha. Usar %c com um espaço antes ignora esses caracteres e lê o próximo caractere válido.

Saída

```c
#include <stdio.h>

int main(){
    int idade;
    float altura;
    char inicial_nome;

    printf("Qual sua idade?\n");
    scanf("%d" , &idade);
    printf("Qual sua altura?\n");
    scanf("%f", &altura);
    printf("Qual sua a letra inicial do seu nome?\n");
    scanf(" %c", &inicial_nome);

    return 0;
}
```

Código 10: Exemplo de saída padrão.

- Em C, é recomendável evitar [3] o uso de acentos e caracteres especiais, como ç e ˜, em identificadores (nomes de variáveis, funções, etc.).
- Caso seja necessário usar caracteres especiais, deve-se utilizar seus códigos correspondentes no padrão ASCII.
- A tabela completa se encontra na Tabela 4 no apêndice.

```c
#include <stdio.h>

int main() {
    //Imprime o cractere é.
    printf("Isso %c trabalhoso!\n", 233);
    return 0;
}
```

Código 11: Exemplo de uso de acento.

[3] Em nossa disciplina, podemos evitar esse complicador, iremos utilizar `eh` em vez do caractere é.

Operadores

```c
#include <stdio.h>
int main(){
    int a, b;
    float resultado;

    printf("Digite um numero:\n");
    scanf("%d", &a);
    printf("Digite outro numero:\n");
    scanf("%d", &b);
    resultado = a + b;
    printf("%d\n", resultado);
    return 0;
}
```

Código 12: Exemplo de operadores.

Operador	Descrição
`+`	Adição
`-`	Subtração
`*`	Multiplicação
`/`	Divisão [4]
`%`	Resto da divisão (Módulo)
`pow()`	Potenciação
`sqrt()`	Radiciação

Tabela 2: Operadores aritméticos.

Uso da função `pow()` da bibioteca `math.h`[5].

```c
#include <stdio.h>
#include <math.h>
int main() {
    int base = 2.0;
    int expoente = 3.0;
    float resultado = pow(base, expoente);
    printf("Resultado: %f \n", resultado);
    return 0;
}
```

Código 13: Exemplo de uso da função `pow()`.

- **( )**
  - Calculado em primeiro lugar.
  - Se houver parênteses aninhados, a expressão dentro do par de parênteses mais interno é calculada primeiro.
  - Se não houver aninhamento, são calculados da esquerda para a direita.
- ***, / ou %**
  - Multiplicação, Divisão, Resto (Módulo): Calculados em segundo lugar.
  - Se houver múltiplos operadores, são calculados da esquerda para a direita.
- **+ ou -**
  - Adição, Subtração: Calculados por último.
  - Se houver múltiplos operadores, são calculados da esquerda para a direita.

Formato	Arredondamento
`%f`	3.33333333333
`%.2f`	3.33
`%.4f`	3.3333

Tabela 3: Formatação de Saída.

Arredondamento.

```c
#include <stdio.h>
#include <math.h>
int main() {
    float numero = 4.5678;
    printf("Número original: %.4f\n", numero);
    return 0;
}
```

Código 14: Exemplo de arredondamento.

[4]Normalmente, uma tentativa de dividir por zero não é definida em sistemas computacionais e em geral resulta em um erro fatal, i.e., um erro que fazcom que o programa seja encerrado imediatamente sem ter sucesso narealização de sua tarefa

[5]Para compilar: `gcc nome_arquivo.c -o nome_arquivo -w -lm`

Constante

```c
#include <stdio.h>
#include <math.h>

#define PI 3.14159265

int main() {
    float area, raio;

    printf("Qual o valor do raio? ");
    scanf("%f", &raio);

    area = PI * pow(raio, 2);
    printf("A área do círculo é: %.2f\n", area);

    return 0;
}
```

Código 15: Exemplo de uso de constante.

Valor de PI definido pela biblioteca `math.h`[6].

```c
#include <stdio.h>
#include <math.h>

int main() {
    float area, raio;

    printf("Qual o valor do raio? \n");
    scanf("%f", &raio);

    area = M_PI * pow(raio, 2);

    printf("A área do círculo é: %.2f\n", area);

    return 0;
}
```

Código 16: Exemplo de uso de constante usando biblioteca `math.h`.

[6]Para compilar: `gcc nome_arquivo.c -o nome_arquivo -w -lm`

Comentários

Comentário de uma linha: Começa com // e vai até o final da linha. **Comentário de múltiplas linhas**: Envolvem o texto com /* ... */.

```c
#include <stdio.h>
int main(){
    //isso é um comentário!
    printf("Hello world!\n");
    //comentário são ignorador pelo compilador!
    //comentários poder ser um lembrete ou uma nota para te ajudar!
    //isso é comentário de uma únicalinha!
    printf("Hello world!\n");
    /* isso é um
    comentários de
    múltiplas linhas */

    return 0;
}
```

Código 17: Exemplo de comentários.

(Deitel; Deitel, 2011) - Capítulo/Seção 2.2, 2.3, 2.4, 2.5.

BLOODSHED. **Dev-C++: Free C++ IDE for Windows**. [*S. l.: s. n.*], 2025. https://bloodshed.net/. Acesso em: 12 jan. 2025.

CODE::BLOCKS. **Code::Blocks: The open source, cross platform, free C++ IDE**. [*S. l.: s. n.*], 2025. https://www.codeblocks.org/. Acesso em: 12 jan. 2025.

DE OLIVEIRA, J.F.; MANZANO, J.A.N.G. **Algoritmos: Lógica para Desenvolvimento de Programação de Computadores**. 16. ed. São Paulo: Editora Érica, 2004.

DE SOUZA, M.A.F. *et al.* **Algoritmos e Lógica de Programação**. São Paulo: Thomson Learning, 2004.

DEITEL, Paul; DEITEL, Harvey. **C: Como Programar**. 6. ed. São Paulo: Pearson Universidades, 2011.

JETBRAINS. **CLion: Cross-platform IDE for C and C++ by JetBrains**. [*S. l.: s. n.*], 2025. https://www.jetbrains.com/clion/. Acesso em: 12 jan. 2025.

MEDINA, M.; FERTIG, C. **Algoritmos e Programação – Teoria e Prática**. São Paulo: Novatec, 2005.

MICROSOFT. **Visual Studio Code**. Acesso em: 5 jan. 2025. 2024. Disponível em: https://code.visualstudio.com/.

Caractere	Código ASCII Decimal
é	233
è	232
á	225
à	224
ç	231
ã	227
õ	245
ó	243
ú	250
#	35
@	64

Tabela 4: Caracteres ASCII.

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

Tabela 5: Palavras Reservadas.

Estes slides estão protegidos por uma licença Creative Commons

Este modelo foi adaptado de Maxime Chupin.

Marisangila Alves, MSc
marisangila.alves@udesc.com
marisangila.com.br

UDESC
UNIVERSIDADE DO ESTADO DE SANTA CATARINA
JOINVILLE
CENTRO DE CIÊNCIAS TECNOLÓGICAS

UDESC
Universidade do Estado de Santa Catarina

2025/1

Algoritmo e Linguagem de Programação

Linguagem de Programação C
Fundamentos