

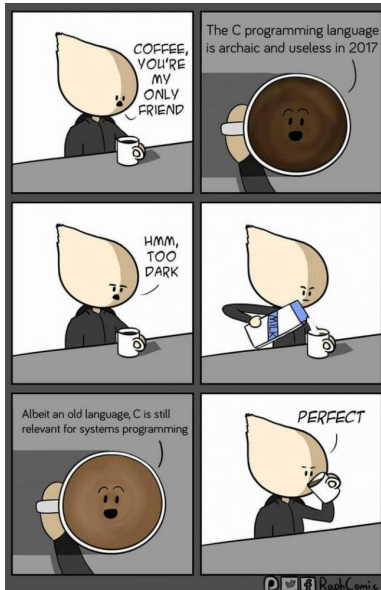
Fundamentos de Programação

Programas em C

Dainf - UTFPR

Profa. Leyza Baldo Dorini
Prof. Bogdan Tomoyuki Nassu

A linguagem C



Características da linguagem C

Algumas das principais características da linguagem C são¹:

- linguagem procedural, modular e estruturada, com tipagem estática de dados
- geralmente é compilada para o código de máquina da plataforma alvo, gerando código compacto e eficiente
- possui facilidades para acesso de baixo nível à memória, registradores e portas de E/S
- extremamente portátil, pode executar em plataformas de microcontroladores a supercomputadores
- é muito utilizada para escrever software de sistema, como sistemas operacionais, compiladores, serviços de rede, interfaces gráficas, bancos de dados, editores de texto, jogos, etc

¹Retirado de <http://wiki.inf.ufpr.br/maziero/doku.php?id=prog2:start>

Um breve histórico

- A linguagem C foi criada por Dennis Ritchie em 1972 para reescrever de forma portátil o sistema operacional UNIX, que antes era escrito em assembly. Sua estrutura e seu nome provêm de uma linguagem anterior B, que era uma simplificação da linguagem de programação BCPL, escrita em 1966.
- Em 1978 foi publicado o famoso livro C Programming Language por Brian Kernighan e Dennis Ritchie, consagrando a linguagem no padrão chamado K&R. Durante muitos anos esse livro foi considerado como a especificação da linguagem.
- Em 1989 o padrão C foi oficializado pelo ANSI (American National Standards Institute) através do padrão ANSI X3.159-1989, conhecido hoje como ANSI C, C89 ou C90. É provavelmente a versão mais usada da linguagem ainda hoje.

Um breve histórico

- Em 1999 a ISO (International Standards Organization) publicou um novo padrão da linguagem chamado ISO/IEC 9899:1999 e conhecido como C99. Em 2011 foi publicado o padrão mais recente, chamado C11.
- Apesar de sua idade, C continua sendo uma das linguagens mais utilizadas hoje em dia (TIOBE INDEX, Ranking IEEE 2018). A linguagem C inspirou a criação de muitas outras linguagens de programação mais recentes, como C++, Java, C#, JavaScript, Perl, PHP, Lua, etc².

²Histórico retirado de

<http://wiki.inf.ufpr.br/maziero/doku.php?id=prog2:start>

Programas em C

Programas em C

Um programa em C é um arquivo texto, contendo declarações e operações da linguagem. Isto é chamado de **código-fonte**.

Exemplo:

```
1  /******  
2  Exemplo de código-fonte da linguagem C  
3  *****/  
4  #include<stdio.h>  
5  
6  void converteHora(int total, int *hora, int *min, int *seg);  
7  
8  int main(){  
9  
10     int input,  
11         h, m, s;  
12  
13     printf("Digite a quantidade de segundos: ");  
14     scanf("%d", &input);  
15  
16     converteHora(input, &h, &m, &s);  
17  
18     printf("%d segundos correspondem a %d horas, %d minutos e %d segundos", input,  
19  
20     return 0;  
21 }
```

Programas em C

A estrutura básica é a seguinte:

```
1  #include <stdio.h>
```

```
2
```

```
3  int main()
```

```
4  {
```

```
5
```

```
6
```

```
7      return 0;
```

```
8  }
```

função principal
(ponto de partida para a
execução do programa)

Programas em C

A estrutura básica é a seguinte:


```
1  #include <stdio.h>
2
3  int main()
4  {
5
6
7      return 0;
8  }
```

o bloco
de comandos
deve ser
delimitado
por chaves

Programas em C

A estrutura básica é a seguinte:

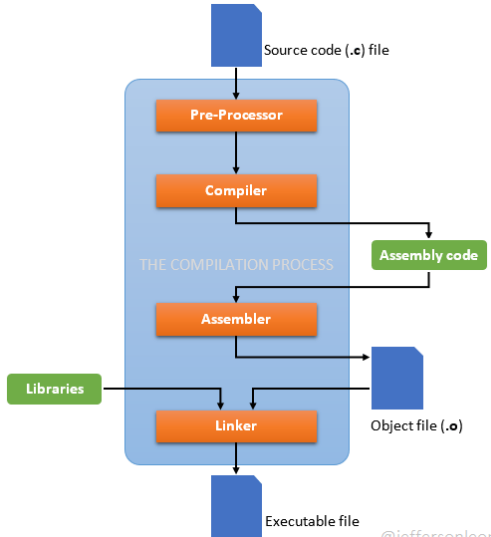
antes da função `main()`, é realizada a inclusão de cabeçalhos da biblioteca padrão da linguagem (`stdio.h`, por exemplo, possui definições de subrotinas relativas a operações de entrada e saída)



```
1  #include <stdio.h>
2
3  int main()
4  {
5
6
7      return 0;
8  }
```

Compile and run

Antes que o programa possa ser executado, ele deve ser **compilado**. Este processo transformará o conjunto de arquivos-fonte em um arquivo executável.



@jeffersonleon

Compilador

Compilador é o programa que, a partir de um código-fonte escrito em C, cria um programa semanticamente equivalente em código de máquina.



A compilação em C envolve 4 fases/estágios:

- Pré-processamento: executa as denominadas diretivas de pré-processamento (em suma, expande as macros e inclui os arquivos de cabeçalho).
- Compilação: nesse momento é realizada a verificação da sintaxe e tradução do código.
- Assembler: o GCC converte o código de máquina de um processador específico em um arquivo objeto.
- Linking: combina o código de máquina gerado com as bibliotecas necessárias, criando um arquivo executável.

A compilação em C envolve 4 fases/estágios:

- Pré-processamento: executa as denominadas diretivas de pré-processamento (em suma, expande as macros e inclui os arquivos de cabeçalho).
- Compilação: nesse momento é realizada a verificação da sintaxe e tradução do código.
- Assembler: o GCC converte o código de máquina de um processador específico em um arquivo objeto.
- Linking: combina o código de máquina gerado com as bibliotecas necessárias, criando um arquivo executável.

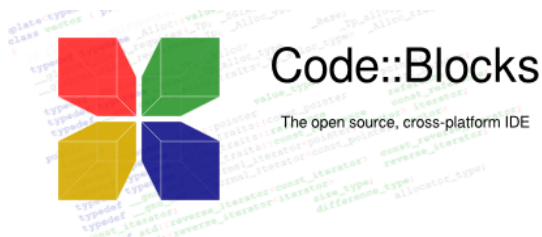
No processo de execução, o arquivo gerado pelo processo de compilação é carregado na memória e a CPU executa instrução por instrução.

Onde escrever os programas?

Code::Blocks

Uma possibilidade é utilizar a IDE (*Integrated Development Environment*) Code::Blocks. Ela une diversas ferramentas de desenvolvimento, dentre as quais:

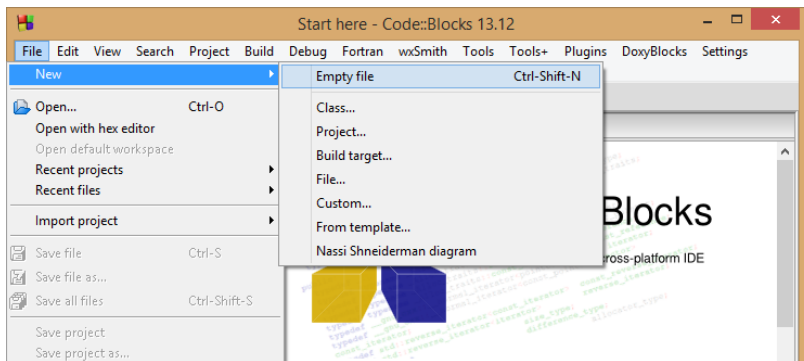
- Um editor (onde podemos digitar o código-fonte)
- Compilador (no caso, o gcc (*Gnu Compiler Collection*))



Apesar de o Code::Blocks apresentar limitações, é *Open Source* (e já está instalado nos laboratórios do DAINF-UTFPR).

Escrevendo o código-fonte

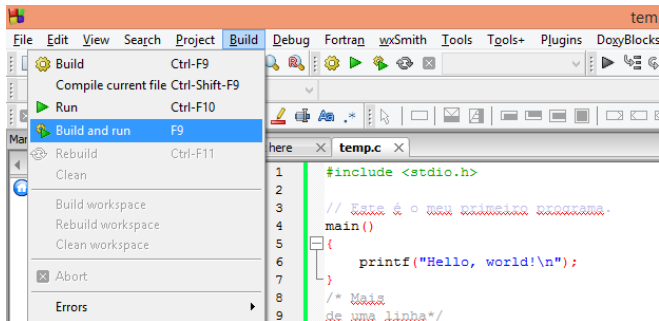
Para criar um arquivo: File - New - Empty File. Lembre-se de salvar com extensão .c.



Obs. Se você não colocar a extensão .c corretamente, não será possível compilar o arquivo. Posteriormente iremos criar projetos. Por enquanto, apenas arquivos.

Compilando e executando o programa no Code::Blocks

Como vimos, para executar um programa a partir do seu código-fonte é necessário compilá-lo e depois executá-lo. No Code::Blocks, a tecla de atalho é F9. Você também pode usar o menu *Build*:



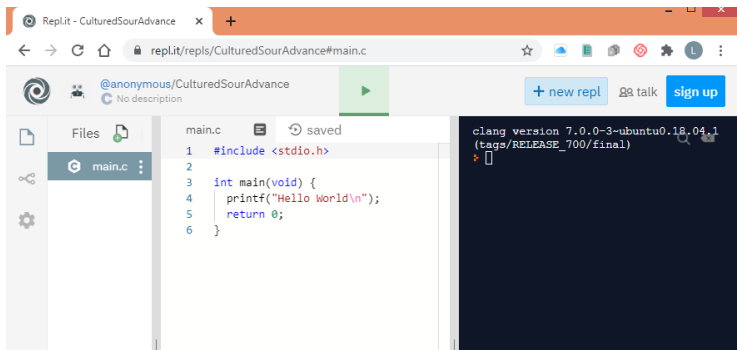
Outra opção consiste em utilizar um editor de texto para escrever o código-fonte (como por exemplo Vim, Emacs, Sublime, etc) e compilar/executar via linha de comando.

```
$ gcc hello.c -o hello
```

```
$ ./hello
```

Ambientes de desenvolvimento online

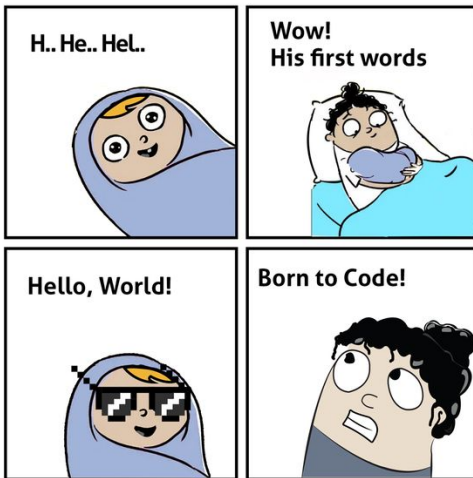
Além disso, também podemos utilizar um ambiente de desenvolvimento online para programar pelo navegador. Um exemplo é o `repl.it`



Compilando programas...



Primeiro programa: Hello World!



Programa básico em C

O nosso primeiro programa em C será o clássico Hello World!³.
Implemente e execute o programa abaixo.

```
1 #include <stdio.h>
2 int main()
3 {
4     printf("Hello, world!");
5     return 0;
6 }
```

³Da Wikipedia: “(...) the tradition of using the phrase Hello, World! as a test message was influenced by an example program in the seminal 1978 book The C Programming Language. The example program in that book prints hello, world, and was inherited from a 1974 Bell Laboratories internal memorandum by Brian Kernighan.”

E os erros?

THE JOYS OF PROGRAMMING™

PHEW , I'VE FINALLY
REDESIGNED AND
CLEANED THE CODE.
LET'S COMPILE.



Compiling failed
42 error(s), 314 warning(s)

IT'S
GOING
TO BE
A LONG
DAY



AFTER ABOUT AN HOUR OF FIXING EACH ERROR ...

Segmentation fault (core dumped).



SHORTLY ...



how to destroy a comp|

how to destroy a computer

how to destroy a computer using no

/spacetime_bender @reddit

O que são erros de **compilação**?

Caso o programa não esteja de acordo com as regras da linguagem, ocorrem erros de compilação, os quais impedem que seja gerado um arquivo executável. **Ler** e entender estes erros é muito importante.

```
1 #include <stdio.h>
2 int main() {
3     printf("Hello, world.\n");
4     return 0;
```

```
$ gcc hello.c -o hello
hello.c: In function `main':
hello.c:7: error: syntax error at end of input
```

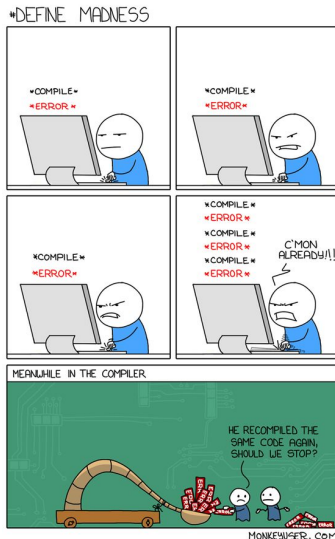
Erros de compilação

Erros de compilação impedem que o arquivo executável seja criado! É preciso corrigi-los!



Erros de compilação

Compilar de forma repetida sem arrumar o código não faz os erros desaparecerem! ;-)



O que são erros de **execução**?

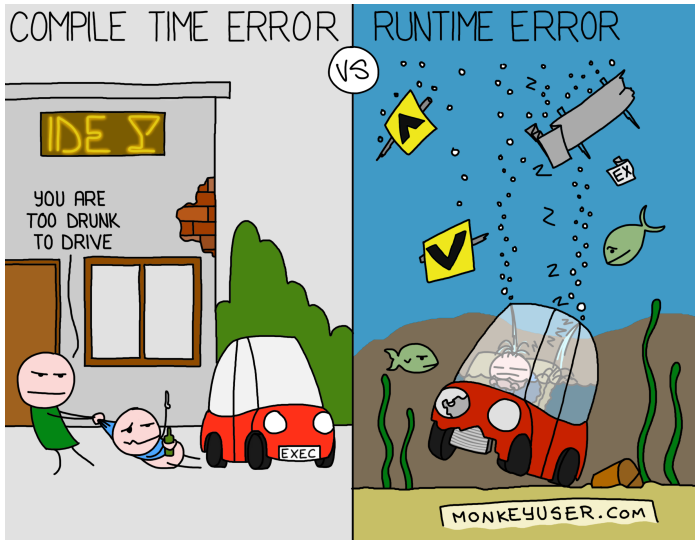
Erros de execução acontecem quando o comportamento do programa diverge do esperado, ou seja, o programa compila mas a saída não é aquela esperada.

```
1 #include <stdio.h>
2 int main() {
3     printf("Hello, world. $#%#@%\n");
4     return 0;
5 }
```

```
$ gcc hello.c -o hello
$ hello
Hello, world. $#%#@%
```

No caso deste exemplo, estamos supondo que os caracteres especiais junto com a mensagem são indesejados.

Para descontrair...



Além dos erros, temos as *warnings*

O compilador gcc também pode indicar *warnings*. Ao contrário dos erros, não impedem que o programa seja compilado e executado.

Embora sejam ignoradas por muitos programadores, devem ser levadas a sério e corrigidas, dado que indicam potenciais inconsistências no código (por exemplo, uso de funções obsoletas, esquecer de incluir headers, uso incorreto de tipos, ...).

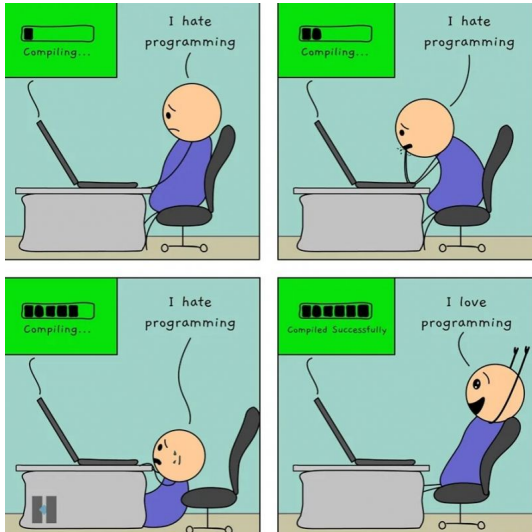
Neste curso, trate *warnings* como erros (ou seja, devem ser corrigidas)!

Debugger (depurador)

Posteriormente, iremos aprender a usar o *debugger*, o qual é utilizado para analisar a execução do programa e localizar erros (*bugs*).



Tarefa



Tarefa!

Sua tarefa consiste nos seguintes itens:

- ① baixar a IDE Code::Blocks (atenção para baixar a versão **com** compilador)
- ② implementar o programa Hello World! (sim, seu primeiro programa deve ter exatamente essa mensagem - afinal, tradição é tradição)
- ③ compilar e executar o programa criado
- ④ implemente novamente o programa em um editor on-line (como o `repl.it`, por exemplo)