UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Fundamentos de Programação

CP41F

Compilação por linha de comando. Ambiente integrado de desenvolvimento (IDE). Parâmetros de compilação. Estrutura de um programa em C. Paradigmas de programação. Operador de atribuição e armazenamento em variáveis.

Aula 4

Prof. Daniel Cavalcanti Jeronymo

Universidade Tecnológica Federal do Paraná (UTFPR)
Engenharia de Computação – 1º Período

Plano de Aula

- Operador de atribuição e armazenamento em variáveis

- Paradigmas de programação

- Estrutura de um programa em C
  - Compilação por linha de comando
  - Parâmetros de compilação
  - Ambiente integrado de desenvolvimento

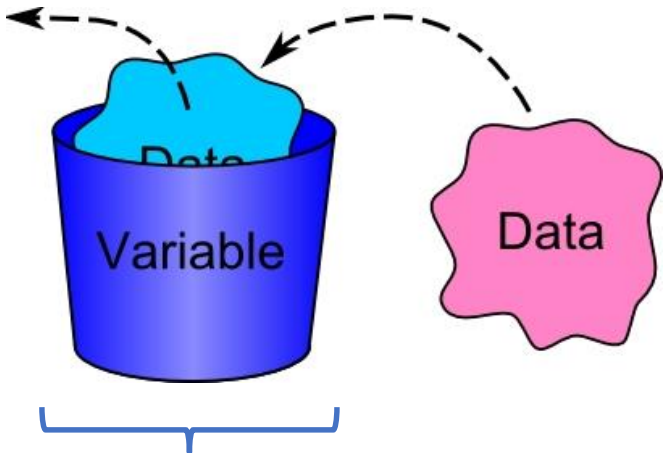
Operador de atribuição e armazenamento em variáveis

- **Variável:**
  - Local de armazenamento associado com um identificador (nome simbólico).
  - O nome permanece fixo porém a informação armazenada varia.
  - Exemplos:

```
tipo variável1;

tipo variável2, Variável2;
```

duas variáveis diferentes!



Operador de atribuição e armazenamento em variáveis

- **Nomes de variáveis:**
  - Devem começar com uma letra ou um sublinhado “_”, não com um número.
  - Pode conter letras, dígitos e “_”
  - Não são permitidos espaços
  - Diferença entre caixa alta e caixa baixa
  - Não pode ser um nome reservado

Nomes reservados:

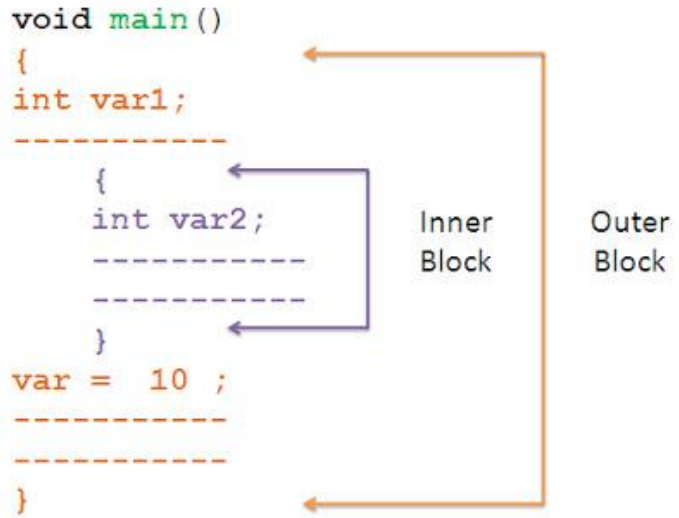
auto	if, int
break	long
case, char, const, continue	register, return
default, do, double	short, signed, sizeof, static, struct, switch
else, enum, extern	typedef
float, for	union, unsigned
goto	void, volatile
	while

Operador de atribuição e armazenamento em variáveis

- **Nomes corretos:**
  - Variavel, VaRiAvEl, nome_da_variável, nome_123, _123
- **Nomes incorretos:**
  - nome da variavel, 123nome, variavel@123, nome-da-variavel, char

Operador de atribuição e armazenamento em variáveis

- **Alcance (visibilidade) de variáveis:**
  - É o escopo de onde a variável pode ser acessada
  - Variáveis declaradas fora de qualquer bloco: global
  - Variável declarada dentro de um bloco: local
  - Variáveis locais podem ser acessadas apenas pelo bloco de origem e seus descendentes



Operador de atribuição e armazenamento em variáveis

- **Alcance (visibilidade) de variáveis:**

```c
int global;
int main(void)
{
    int local;                          /* variável local */

    global = 1;                         /* global pode ser usada aqui */
    local = 2;

    {
        int muito_local = global + local;  /* variável local ao bloco */
    }

    /* muito_local não pode ser usada aqui*/

    return 0;
}
```

Operador de atribuição e armazenamento em variáveis

- **Operador de atribuição:**
  - Atribui valores a variáveis.
  - Sintaxe:

    variável = valor;

  - Exemplos:

```
int a;
int b = 0;
int c = 2 + 5;
a = 10;
```

variável NÃO inicializada

variáveis inicializadas

Operador de atribuição e armazenamento em variáveis

- **Operador de atribuição:**
  - Pode conter expressões no lado direito

    variável = expressão;

  - A expressão é calculada e o valor resultante é atribuído a variável

Operador de atribuição e armazenamento em variáveis

- **Operador de atribuição:**
  - O sinal de igual é o operador de atribuição
  - Outros operadores serão vistos futuramente
  - Sintaxe geral:

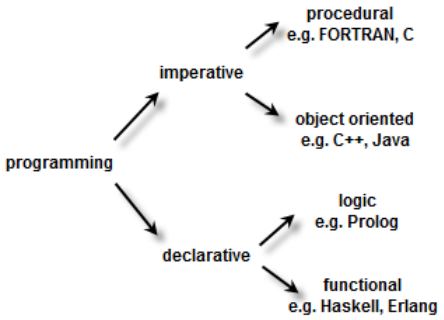
    lvalue = *rvalue*

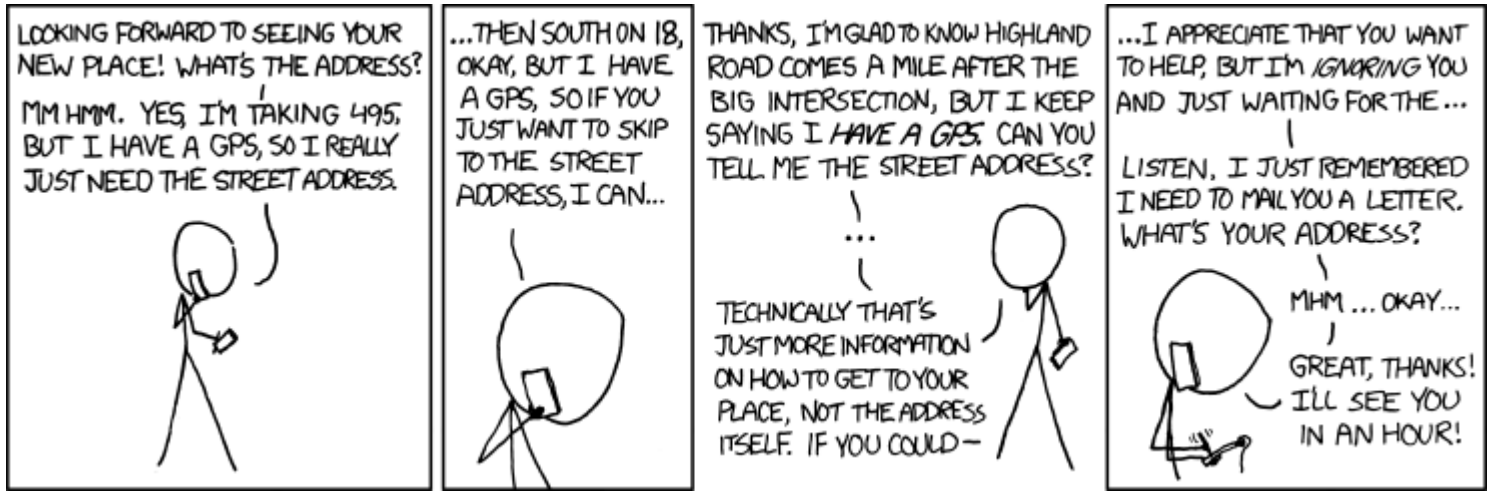
    lvalue é uma variável

    rvalue é uma expressão, constante ou variável

Paradigmas de Programação

- **Paradigmas principais:**
  - Declarativa (*what*) x Imperativa (*how*)





  - Não-estruturada x Estruturada

The Evolution of Expression

A structured `while` *loop is easier to read than the convoluted* `goto` *approach.*

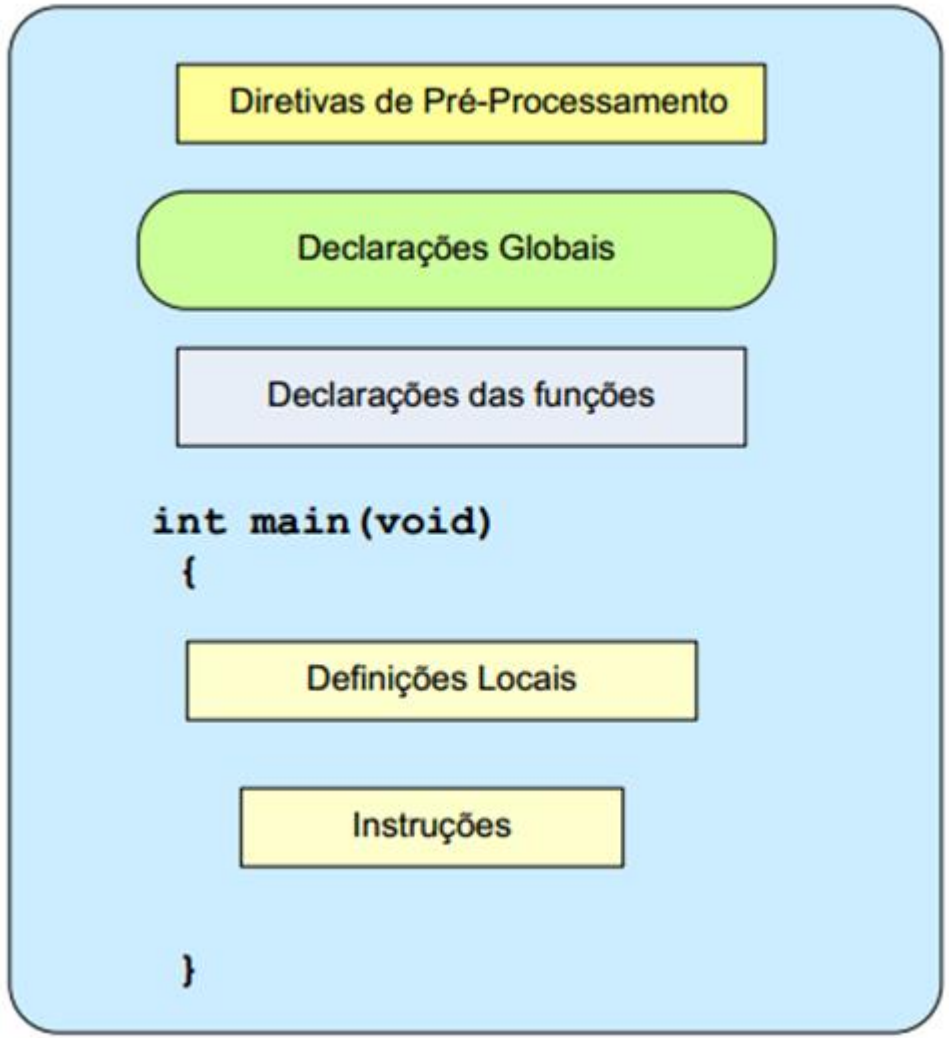
Instead of writing ...

```
i = 1
START: if i = 4
  then goto END
    print(i)
    i = i + 1
    goto START
END:
```

we write ...

```
i = 1;
while (i < 4) {
  print(i);
  i = i + 1;
}
```

Estrutura de um programa em C




```c
#include <stdio.h>

int main(void)
{
    printf("Ola mundo!\n");

    return 0;
}
```

Compilação por linha de comando

- **Escreva o seguinte programa e salve-o como test.c:**

```
#include <stdio.h>

int main(void)
{
   printf(“Ola mundo!\n”);
   return 0;
}
```

Compilação por linha de comando

- **Abra o console (shell ou linha de comando) na pasta onde o arquivo test.c foi salvo e digite:**

```
gcc.exe test.c –o test.exe
```

- **!ERRO! No caso de erro afirmando que gcc.exe não foi encontrado, é necessário colocar a pasta do compilador no PATH.**

Compilação por linha de comando

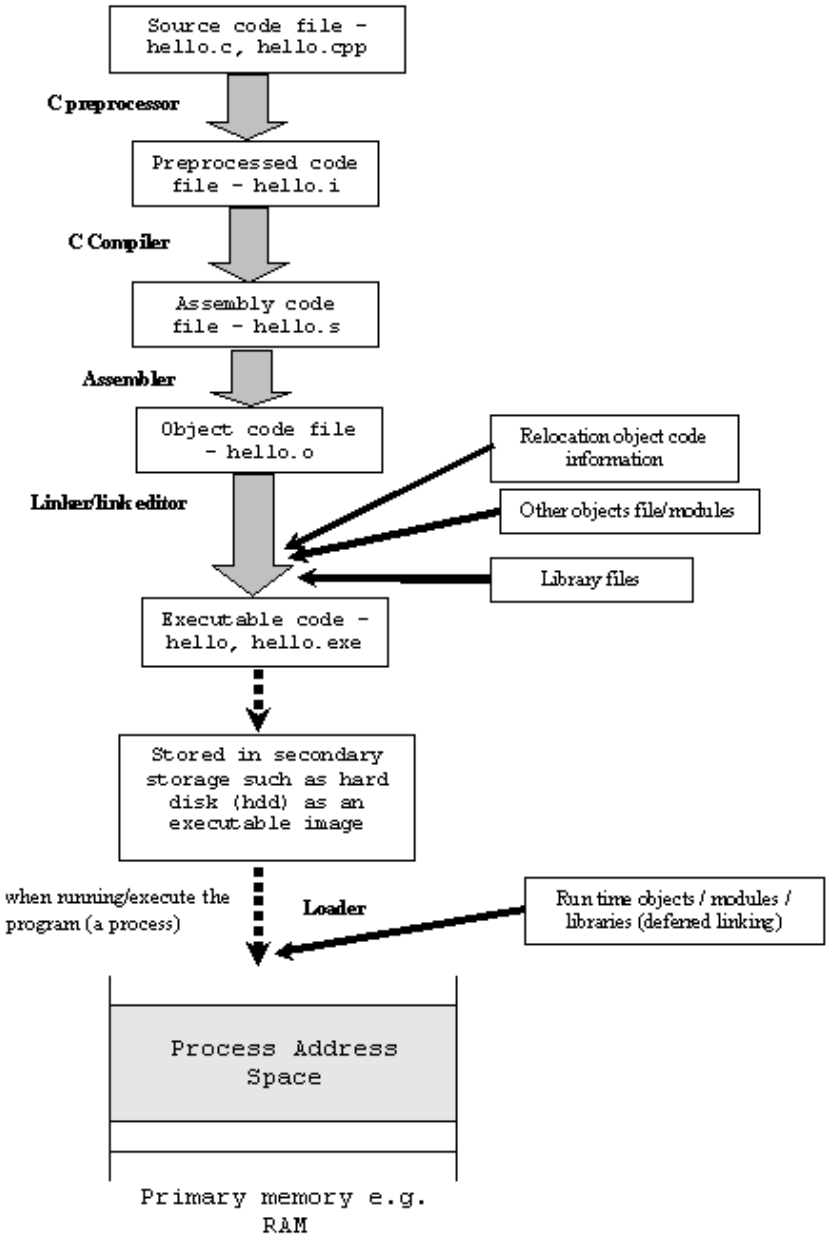
- **Para ajustar o PATH digite no console:**

```
set PATH=%PATH%;C:\Program Files (x86)\CodeBlocks\MinGW\bin
```

- **Essa linha de PATH assume sistema operacional Windows x64 e CodeBlocks instalado com MinGW.**

Parâmetros de compilação

- **Conseguiu compilar?**
  - Teste parâmetros diferentes do compilador: -E -S –c –o
  - Compare os resultados obtidos com o esperado do ciclo de compilação.
  - Na dúvida sobre cada parâmetro, procure referências do compilador!



Parâmetros de compilação

- **Dessa vez utilize os seguintes parâmetros:**

```
gcc.exe -pedantic-errors -Wextra -Wall -ansi test.c –o test.exe
```

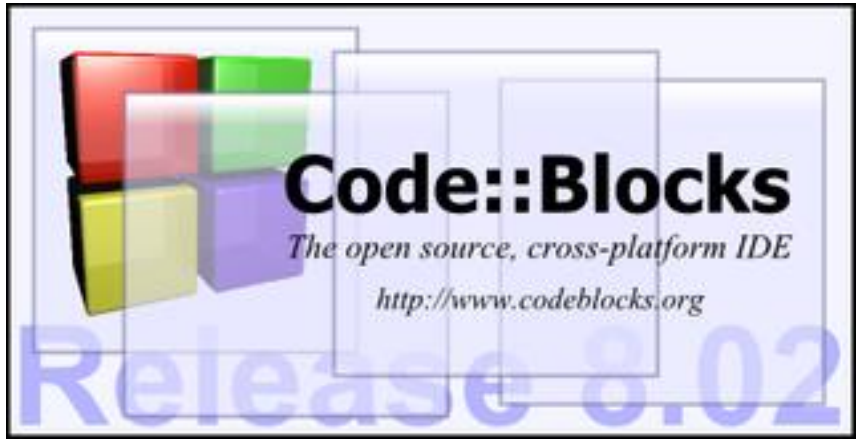
- **Leia no manual do GCC o que cada parâmetro faz!**

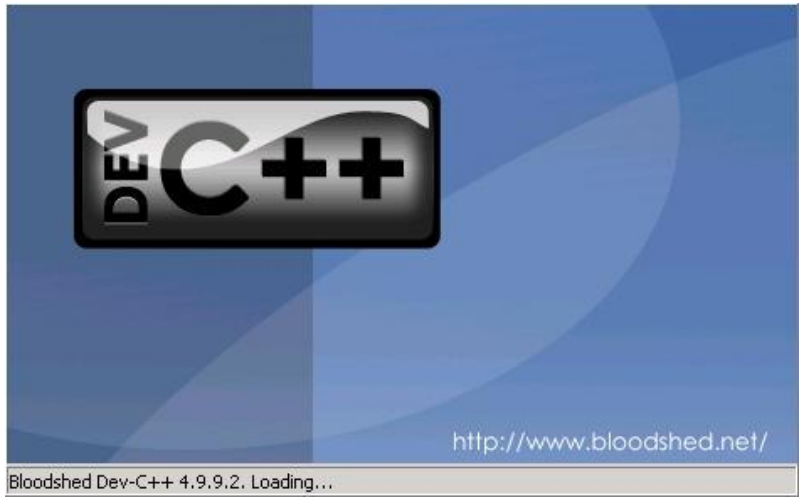
https://gcc.gnu.org/onlinedocs/gcc-3.0.4/gcc_3.html

https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html

Ambiente de Desenvolvimento

- **Repita o processo de compilação.**
  - Dessa vez utilize uma IDE (*Integrated Development Environment*) da sua escolha





Visual Studio®

Exercícios

- **Exercício 1 – Qual o valor armazenado em a ao final do programa?**

```
int main(void)
{
   int a, b=10, c, d;
   d = 1;
   c = 5;
   d = c + b;
   a = c + 2;
   a = a + 1;
}
```

Exercícios

- **Exercício 2 – Implemente o algoritmo de solução do problema dos galões. Qual a saída de g5 e de g3?**

```
int main(void)
{
   int g5 = 0, g3 = 0;
   .
   .
   .
   printf("%d %d\n", g5, g3);
   return 0;
}
```