

Programação e Desenvolvimento de Software I

Variáveis

Prof. Héctor Azpúrua
(slides adaptados do Prof. Pedro Olmo)

Sobre honestidade acadêmica



Honestidade Acadêmica

Comportar-se com honestidade acadêmica significa que a essência de todos os trabalhos que você enviar para este curso deve ser sua. Você pode discutir e tirar dúvidas com outros colegas de turma, entretanto, sugerimos que **não haja compartilhamento de código.**

Uso de IA Generativa (ChatGPT, etc)

"Se você deixar a IA te substituir..."

Evite usar...

Lembre-se, a sua reputação profissional já começa a ser construída desde agora.

...atividades será feita uma análise de plágio (automática e manual). Trabalho

forem considerados com alto nível de semelhança serão zerados.

Lembre-se, a sua reputação profissional já começa a ser construída desde agora.

Links sobre honestidade acadêmica:

- Dicas do Prof. Fernando: [aqui](#)
- Regras acadêmicas - UFMG: [aqui](#)



Repasso da aula anterior

Duvidas

- Qual é a diferença entre esses dois códigos?
 - (os dois vão compilar)

```
int minha_funcao() {  
    printf("ola!");  
    return 0;  
}
```

```
int minha_funcao() {  
    printf("ola!");  
}
```

Mas um tipo de retorno
dessa função foi **SIM** declarado

Não usa return

- O que esta acontecendo? 🤔
 - Seção 6.9.1p12 do padrão de C (livro)

12 If the } that terminates a function is reached, and the value of the function call is used by the caller, the behavior is undefined.

Repasso da aula anterior

Duvidas

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

```
// gcc -o main main.c
```

```
int func_int() {
}
```

```
char func_char() {
}
```


```
float func_float() {
}
```

```
int main() {
    printf("%d\n", func_int());
    printf("%c\n", func_char());
    printf("%f\n", func_float());

    return 0;
}
```

■ Qual é a saída desse código?

```
● (python3.12_venv) → no_return gcc -o main main.c && ./main
main.c:7:1: warning: non-void function does not return a value [-Wreturn-type]
    7 | }
      | ^
main.c:10:1: warning: non-void function does not return a value [-Wreturn-type]
   10 | }
      | ^
main.c:13:1: warning: non-void function does not return a value [-Wreturn-type]
   13 | }
      | ^
3 warnings generated.
1
0.000000
```



Repasso da aula anterior

Duvidas

- Qual é a diferença entre esses dois códigos?

```
int minha_funcao() {  
    int num = 123;  
    printf("numero: %d", num);  
    return 0;  
}
```

```
int minha_funcao() {  
    int num = 123;  
    printf("numero: %i", num);  
    return 0;  
}
```

- O que esta acontecendo? 🤔
 - <https://en.cppreference.com/w/c/io/fprintf>

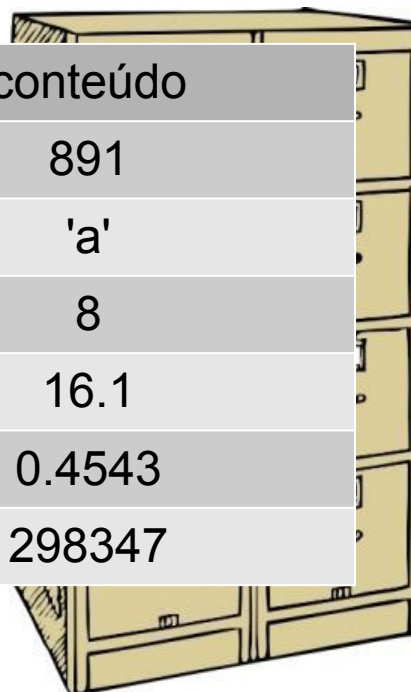
d i	Converts a signed integer into decimal representation <code>[-]dddd</code> . <ul style="list-style-type: none">• <i>Precision</i> specifies the minimum number of digits to appear. The default precision is <code>1</code>.• If both the converted value and the precision are <code>0</code> the conversion results in no characters.• For the z modifier, the expected argument type is the signed version of <code>size_t</code>.
----------------------	---

Variáveis

O que são?

- Os dados que um programa utiliza precisam ser armazenados na **memória do computador**
- Cada posição de memória do computador possui um **endereço**

endereço		endereço	conteúdo
Rua do Ouro, 12		6612	891
Rua do Ouro, 13		6613	'a'
Rua do Ouro, 14		6614	8
Rua do Ouro, 15		6615	16.1
Rua do Ouro, 16		6616	0.4543
Rua do Ouro, 17		6617	298347



Cada gaveta tem uma etiqueta e um espaço bem delimitado. No entanto, você pode guardar **diversas coisas dentro delas**.

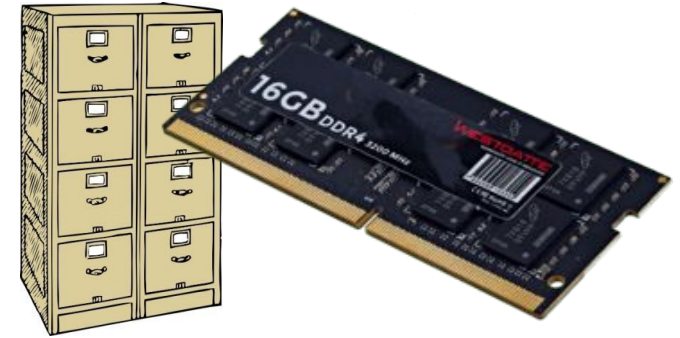
Variáveis

O que são?

- A partir dos endereços, é possível para o computador saber qual é o valor armazenado em cada uma das posições de memória
- Como a **memória pode ter bilhões de posições**, é difícil controlar em qual endereço está armazenado um determinado valor!
- Para facilitar o controle sobre onde armazenar informação, os programas utilizam **variáveis**:
 - Uma variável corresponde a um **nome simbólico (ou etiqueta)** de uma posição de memória
 - Seu **conteúdo pode variar** durante a execução do programa

Variáveis

Representação no computador



Variável	Endereço
idade	6614
salario	6612
frac	6615

Variável usada no
programa

Endereço de memória em
que o conteúdo da variável
está armazenado

Endereço	Conteúdo
6611	9439.23496
6612	891
6613	'P'
6614	8
6615	0.4543
6616	2365

Variáveis

Representação no computador

- Memória + dicionário de variáveis
 - Vamos usar esta representação ao longo do curso!

Endereço	Variável	Conteúdo
6612	salario	891
6613	c	'a'
6614	idade	8
6615	velocidade	16.1
6616	frac	0.4543
6617	km	298347

Variáveis

programa1.c

- Exemplo de variável no nosso programa1.c:
 - Onde tem a variável?

```
1  #include <stdio.h>
2  #include <math.h>
3
4  int main(int argc, char* argv[]) {
5      float y;
6      y = sin(1.5);
7      printf("seno de 1.5 eh: %f", y);
8      printf("\n");
9      system("PAUSE");
10     return 0;
11 }
```

A variável `y` irá armazenar o valor de `sin(1.5)`

Variáveis

- **Cada variável pode possuir uma quantidade diferente de bytes**, uma vez que os tipos de dados são representados de forma diferente
- Portanto, a cada variável está associado um tipo específico de dados
- Logo:

O tipo da variável define quantos bytes de memória serão necessários para **representar os dados** que a variável armazena

Variáveis

Tipos de dados

- A Linguagem C dispõe de **quatro tipos básicos de dados**
- Assim, as variáveis poderão assumir os seguintes tipos:

Tipo	Tamanho (bytes)	Valor
char	1	Um caractere ou um inteiro de 0 a 127
int	4	Um número inteiro
float	4	Um número de ponto flutuante (SP)
double	8	Um número de ponto flutuante (DP)

Variáveis

Declaração

- Dentro do programa, as variáveis são identificadas por seus **nomes**
- Portanto, um programa deve **declarar todas as variáveis** que irá utilizar
- Atenção!
 - A **declaração de variáveis deve ser feita antes** que a variável seja usada, para garantir que a quantidade correta de memória já tenha sido reservada para armazenar seu valor

Variáveis

Assinalar

- Para assinalar valores à variáveis deve-se usar:
 - O operador de atribuição “=”

`nome_da_variável = (expressão) ;`

```
float c;  
c = 0.01;  
int a = 9;  
float b = 0.1;  
char letra = 'a';  
float dinheiro = a * b;
```

Variáveis

Exemplo

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

```
int main() {  
    int idade;  
    float salario;  
    char sexo;  
    double divida;  
  
    idade = 25;  
    salario = 100.5;  
    sexo = 'M';  
    divida = 29999.99;  
  
    printf("Eu tenho %d anos", idade);  
    printf(" recebo %f reais por mes", salario);  
    printf(" sou do sexo %c", sexo);  
    printf(" e tenho uma divida de %f", divida);  
    printf("\n");  
    getchar(); // parar a execucao  
    return 0;  
}
```

Endereço	Variável	Conteúdo
4812		
4813		
4814		
4815		
4816		
4817		
4818		
4819		

Variáveis

Exemplo

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

```
int main() {
```

```
    int idade;  
    float salario;  
    char sexo;  
    double divida;
```

```
    idade = 25;  
    salario = 100.5;  
    sexo = 'M';  
    divida = 29999.99;
```

```
    printf("Eu tenho %d anos", idade);  
    printf(" recebo %f reais por mes", salario);  
    printf(" sou do sexo %c", sexo);  
    printf(" e tenho uma divida de %f", divida);  
    printf("\n");  
    getchar(); // parar a execucao  
    return 0;  
}
```

Endereço	Variável	Conteúdo
4812	idade	
4813	salario	
4814	sexo	
4815	divida	
4816		
4817		
4818		
4819		

Variáveis

Exemplo

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

```
int main() {  
    int idade;  
    float salario;  
    char sexo;  
    double divida;
```

```
    idade = 25;  
    salario = 100.5;  
    sexo = 'M';  
    divida = 29999.99;
```

```
    printf("Eu tenho %d anos", idade);  
    printf(" recebo %f reais por mes,", salario);  
    printf(" sou do sexo %c", sexo);  
    printf(" e tenho uma divida de %f", divida);  
    printf("\n");  
    getchar(); // parar a execucao  
    return 0;  
}
```

Endereço	Variável	Conteúdo
4812	idade	25
4813	salario	100.5
4814	sexo	'M'
4815	divida	29999.99
4816		
4817		
4818		
4819		

Variáveis

programa1.c

- A primeira linha do corpo da função principal no nosso `programa1.c`:
 - É uma declaração de variável

```
1  #include <stdio.h>
2  #include <math.h>
3
4  int main(int argc, char* argv[]) {
5      float y;
6      y = sin(1.5);
7      printf("seno de 1.5 eh: %f", y);
8      printf("\n");
9      system("PAUSE");
10     return 0;
11 }
```

Variáveis

Declaracões

```
float y;
```

- Esta linha declara uma variável `y` para armazenar um número de ponto flutuante *Single-Precision* (SP)
- **A declaração de uma variável não armazena** valor algum na posição de memória que a variável representa
- Ou seja, no caso anterior, vai existir uma posição de memória chamada `y`
 - mas ainda **não vai existir valor armazenado nesta posição**

Variáveis

Múltiplas declarações

- Podemos declarar duas ou mais variáveis no mesmo tipo numa mesma linha, separados por vírgulas:

```
tipo variável_1, variável_2, variável_3;
```

```
float y, aux, salario;
```

```
int x, juros, num_carros;
```

```
char sexo, s;
```

Escrevendo um programa em C

- Um valor pode ser atribuído a uma posição de memória representada por uma variável pelo operador de **atribuição** “=”
- O operador de atribuição requer:
 - À **esquerda** um nome de variável e à **direita**, um valor
 - `variavel = valor;`
- A linha seguinte de `programa1.c` atribui um valor a `y`:

```
1  #include <stdio.h>
2  #include <math.h>
3
4  int main(int argc, char* argv[]) {
5      float y;
6      y = sin(1.5);
7      printf("seno de 1.5 eh: %f", y);
8      printf("\n");
9      system("PAUSE");
10     return 0;
11 }
```

Escrevendo um programa em C

- No lado direito do operador de atribuição “=” existe uma referência à função `seno` com um parâmetro `1.5`
 - Uma **constante de ponto flutuante** representando um valor em radianos

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

```
y = sin(1.5);
```

```
9 system("PAUSE");
10 return 0;
11 }
```

Escrevendo um programa em C

- Em uma linguagem de programação chamamos o valor entre parênteses da função, no exemplo anterior, o valor `1.5`, de **parâmetro da função**

```
void funcao(float parametro1, int parametro2) {  
    // corpo da funcao  
}
```

- Da mesma forma, diz-se que `sin(1.5)` é o valor da função `sin` para o parâmetro `1.5`

```
float sin(float parametro1) {  
    // corpo da funcao  
}
```

- O operador de atribuição na linha `y = sin(1.5)` obtém o valor da função (`0.997495`) e o armazena na posição de memória identificada pelo nome `y`
 - Esta operação recebe o nome de: **atribuição de valor a uma variável**

Escrevendo um programa em C

- **Atenção:** O valor armazenado em uma variável por uma operação de atribuição **depende do tipo da variável**
- Se o tipo da variável for `int`, será armazenado um valor inteiro
 - Caso o valor possua parte fracionária, ela será desprezada:

```
float a = 1.2;  
int b = a; // b vai ter o valor 1 ao invés de 1.2
```

- Se o tipo da variável for `float` ou `double`, será armazenado um valor de ponto flutuante
 - Caso o valor não possua parte fracionária, ela será nula:

```
int c = 9;  
float f = c; // f vai ter o valor 9.0
```


Escrevendo um programa em C

Exemplo

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

int main() {
    float y;
    y = sin(1.5);
    printf("seno de 1,5 eh: %f, y");
    printf("\n");
    getchar();
    return 0;
}
```

Endereço	Variável	Conteúdo
4812		
4813		
4814		
4815		
4816		
4817		
4818		
4819		

Escrevendo um programa em C

Exemplo

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

int main() {
    float y;
    y = sin(1.5);
    printf("seno de 1,5 eh: %f, y");
    printf("\n");
    getchar();
    return 0;
}
```

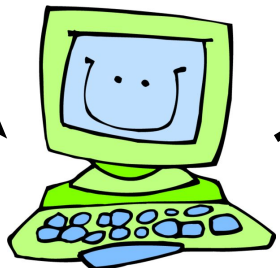
Endereço	Variável	Conteúdo
4812	y	
4813		
4814		
4815		
4816		
4817		
4818		
4819		

Escrevendo um programa em C

Exemplo

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

int main() {
    float y;
    y = sin(1.5);
    printf("seno de 1,5 eh: %f, y");
    printf("\n");
    getchar();
    return 0;
}
```



Endereço	Variável	Conteúdo
4812	y	
4813		
4814		
4815		
4816		
4817		
4818		
4819		

O computador processa a função $\sin(1.5)$

Escrevendo um programa em C

Exemplo

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

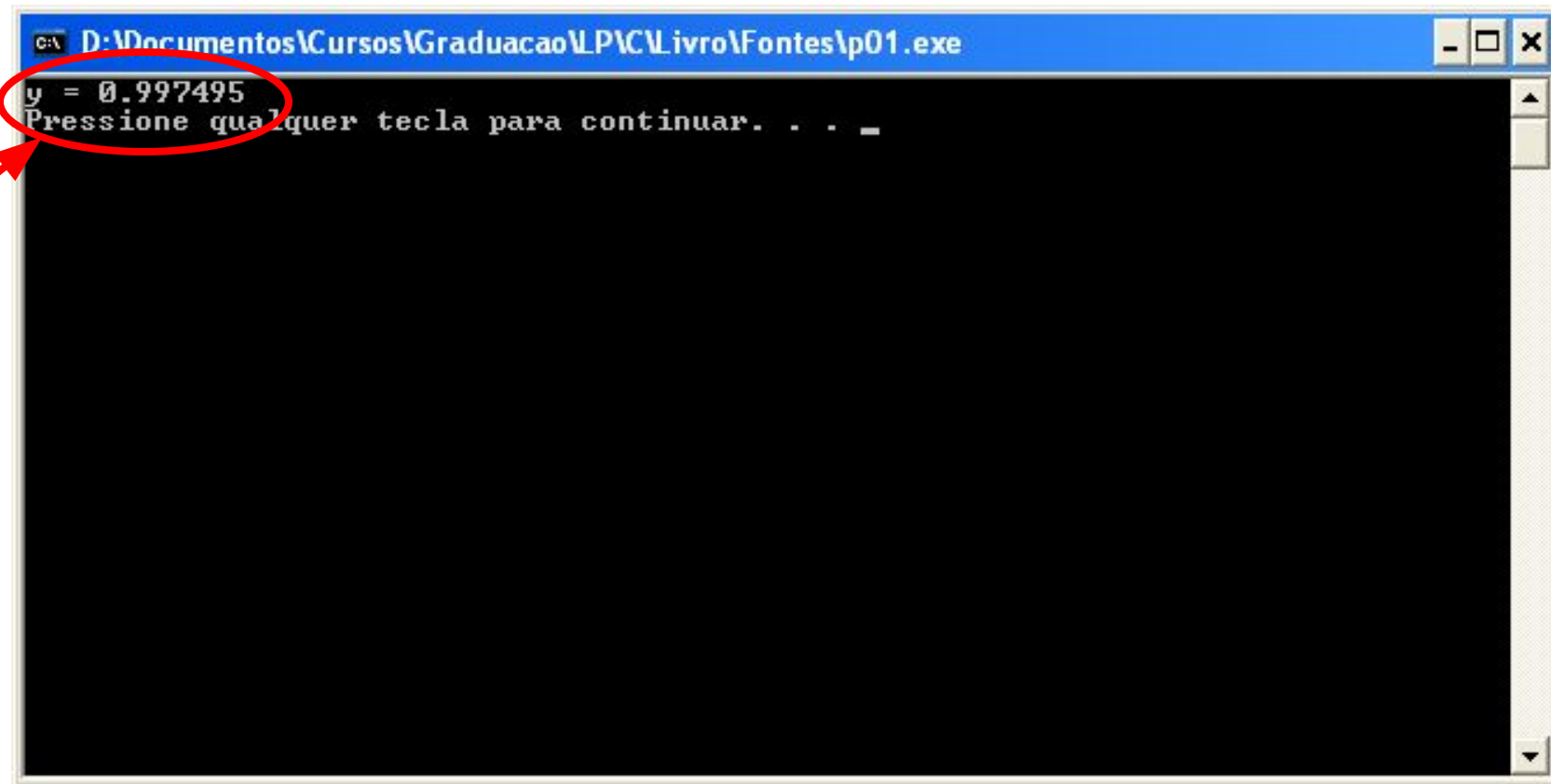
int main() {
    float y;
    y = sin(1.5);
    printf("seno de 1,5 eh: %f", y);
    printf("\n");
    getchar();
    return 0;
}
```

- A função printf faz parte da biblioteca `stdio.h`
- Essa sequência de caracteres pode conter algumas **tags** que representam valores, conhecidas como especificadores de formato
- Um especificador de formato começa sempre com o símbolo “%”
 - Em seguida, pode apresentar uma letra que indica o tipo do valor a ser exibido

Escrevendo um programa em C

Mostrando o resultado no terminal

- Veja:



Valor
armazenado
em *y*

```
printf("y = %f", y);
```

Escrevendo um programa em C

Tags do printf

- Na função `printf`, para cada `tag` existente no primeiro parâmetro, deverá haver um novo parâmetro que especifica o valor a ser exibido

```
int a = 9;  
float b = 0.1;  
printf("a=%d, b=%c e c=%f", a, 'm', (a+b));
```

Escrevendo um programa em C

Tags do printf

- Além de especificar o número de casas decimais, um **tag** pode especificar o número total de caracteres (incluindo o sinal e o ponto decimal)
- Assim, o **tag** `%8.3f` significa:
 - “exibir um valor de **ponto flutuante** com oito caracteres no total e com três casas decimais”
 - Se for necessário, será acrescentado o caractere ‘ ’ (espaço) à esquerda do valor para completar o tamanho total

Escrevendo um programa em C

Tags do printf

- Exemplos:

Valor	Tag	Valor exibido
pi = 3.14159	%5.3f	3.142
	%8.3f	3.142
raio = 2.0031	%5.3f	2.003
	%.6f	2.003100
r2 = 1.25856	%5.3f	12.586
	%6.3f	12.586
	%7.3f	12.586
	%e	1.258560e+00
	%E	1.258560E+00
	%12.3e	1.259e+00

Escrevendo um programa em C

Tags do printf

- A formatação de valores pode ser feita também para **números inteiros**
- Exemplos:

Valor	Tag	Valor exibido
3	%d	3
	%5d	3
	%01d	3
	%05d	00003

Escrevendo um programa em C

Tags do printf

- A lista inteira de formatação do `printf` pode ser acessada na referência de C
 - <https://en.cppreference.com/w/c/io/fprintf>
- Exemplos:

The following format specifiers are available:

Conversion Specifier	Explanation	Expected Argument Type								
		hh	h	none	l	ll	j	z	t	L
		Yes				Yes	Yes	Yes	Yes	
	Only available since C99→	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
%	Writes literal %. The full conversion specification must be %%. Writes a single character . <ul style="list-style-type: none">• The argument is first converted to <code>unsigned char</code>.• If the <code>l</code> modifier is used, the argument is first converted to a character string as if by <code>%ls</code> with a <code>wchar_t[2]</code> argument.	N/A	N/A	int	wint_t	N/A	N/A	N/A	N/A	N/A
c	Writes a character string . <ul style="list-style-type: none">• The argument must be a pointer to the initial element of an array of characters.• <i>Precision</i> specifies the maximum number of bytes to be written. If <i>Precision</i> is not specified, writes every byte up to and not including the first null terminator.• If the <code>l</code> specifier is used, the argument must be a pointer to the initial element of an array of <code>wchar_t</code>, which is converted to <code>char</code> with zero-initialized	N/A	N/A	char*	wchar_t*	N/A	N/A	N/A	N/A	N/A

Escrevendo um programa em C

- A linguagem C utiliza o símbolo \ (barra invertida) para especificar alguns caracteres especiais:

Caractere	Significado
\a	Caractere (invisível) de aviso sonoro.
\n	Caractere (invisível) de nova linha.
\t	Caractere (invisível) de tabulação horizontal.
\'	Caractere de apóstrofo

Escrevendo um programa em C

- Observe a linha 8 do programa `programa1.c`:
- Ela exibe “o caractere (invisível) de nova linha”
 - Qual o efeito disso? Provoca uma mudança de linha! Próxima mensagem será na próxima linha

`printf("\n");`



```
1  #include <stdio.h>
2  #include <math.h>
3
4  int main(int argc, char* argv[]) {
5      float y;
6      y = sin(1.5);
7      printf("seno de 1.5 eh: %f", y);
8      printf("\n");
9      system("PAUSE");
10     return 0;
11 }
```

Avaliação de expressões aritméticas

- Os operadores aritméticos disponíveis na linguagem C são:

Operador	Operação
+	soma
-	subtração
*	multiplicação
/	divisão
%	resto da divisão ou módulo

Conversão implícita de tipo

- Na avaliação de expressões aritméticas, estas **operações são realizadas sempre entre operandos de mesmo tipo**
 - Ou seja, o resultado da operação terá o mesmo tipo que os operandos
- Caso haja valores inteiros e em ponto flutuante em uma expressão, haverá uma conversão implícita de tipo de `int` para `float`, sempre que necessário
- Porque as operações aritméticas devem ser feitas entre operandos do mesmo tipo?
 - As representações dos números inteiros e dos números de ponto flutuante são diferentes
 - **Ou seja, embora 1 e 1.0 são valores iguais, eles têm representações diferentes no computador**

Prioridade de execução das operações

- A execução de operações segue uma ordem específica que **sempre é cumprida**:
 - 1) Expressões entre parênteses
 - 2) Multiplicação, divisão e resto da divisão (da esquerda para a direita)
 - 3) Operações de soma e subtração (da esquerda para a direita)



Prioridade de execução das operações

Exemplo

- 1) Expressões entre parênteses
- 2) Multiplicação, divisão e resto da divisão
- 3) Operações de soma e subtração

▪ Exemplo: $a = 1.5$, $b = 4$, $c = 2$, $d = 3$, $e = 1.2$, $f = 4.5$

$$v1 = (a * (c + d)) / (b * (e + f));$$

Ordem	Operação	Resultado	Conversão de tipo
-------	----------	-----------	-------------------

Conversão implícita de tipo

- É preciso **muito cuidado com a divisão inteira**
 - (divisão entre operandos inteiros)
- O resultado da divisão inteira é sempre um número inteiro
 - Assim, se necessário, pode-se usar uma **conversão explícita de tipo (type casting)**

```
int a = 10, b = 3;
```

```
int c;
```

```
float d;
```

```
c = a / b;                      → c = 3
```

```
d = (float) a / b;              → d = 3.333333
```

Conversão implícita de tipo

- **Atenção!** Observe que os resultados dos seguintes códigos são diferentes!

`d = (float) a / b;`

`d = (float) (a / b);`

- O que tem de diferente?
- No primeiro:
 - Primeiro realiza-se primeiro a conversão explícita de tipo (`a` torna-se `10.0`) e, em seguida, realiza-se a divisão
 - Logo: `d = 3.333333`
- No segundo:
 - Primeiro divide-se `a` por `b` e, em seguida, se faz a conversão explícita de tipo
 - Logo: `d = 3.0`

Realizando operações

Soma, resta, multiplicação e divisão

- Exemplo: Considere as seguintes declarações e operações:

```
int a;  
float b;
```

Operação de atribuição	Valor armazenado
$a = (2 + 3) * 4$	

Realizando operações

Modulo

- Exemplo: Considere as seguintes declarações e operações com módulo:
 - Modulo só funciona com inteiros

`int a;`

`float b = 100.0;`

Operação de atribuição	Valor armazenado
<code>a = 18 % 5</code>	

Escrevendo um programa em C

Exemplo

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

```
int main() {
```

```
    float n;
```

```
    n = 10.0;
```

```
    n = n * 1.1;
```

```
    n = n / 2;
```

```
    int a = n;
```

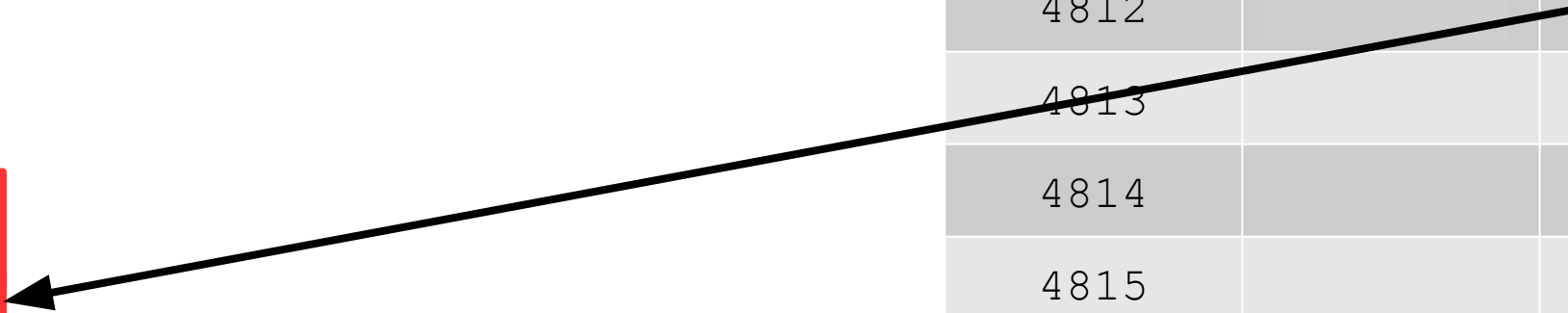
```
    printf("a = %d, n=%f", a, n);
```

```
    getchar();
```

```
    return 0;
```

```
}
```

Endereço	Variável	Conteúdo
4812		
4813		
4814		
4815		
4816		
4817		
4818		
4819		



Escrevendo um programa em C

Exemplo

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

```
int main() {  
    float n;  
    n = 10.0;  
    n = n * 1.1;  
    n = n / 2;  
    int a = n;  
    printf("a = %d, n=%f", a, n);  
    getchar();  
    return 0;  
}
```

Endereço	Variável	Conteúdo
4812	n	11.0
4813		
4814		
4815		
4816		
4817		
4818		
4819		



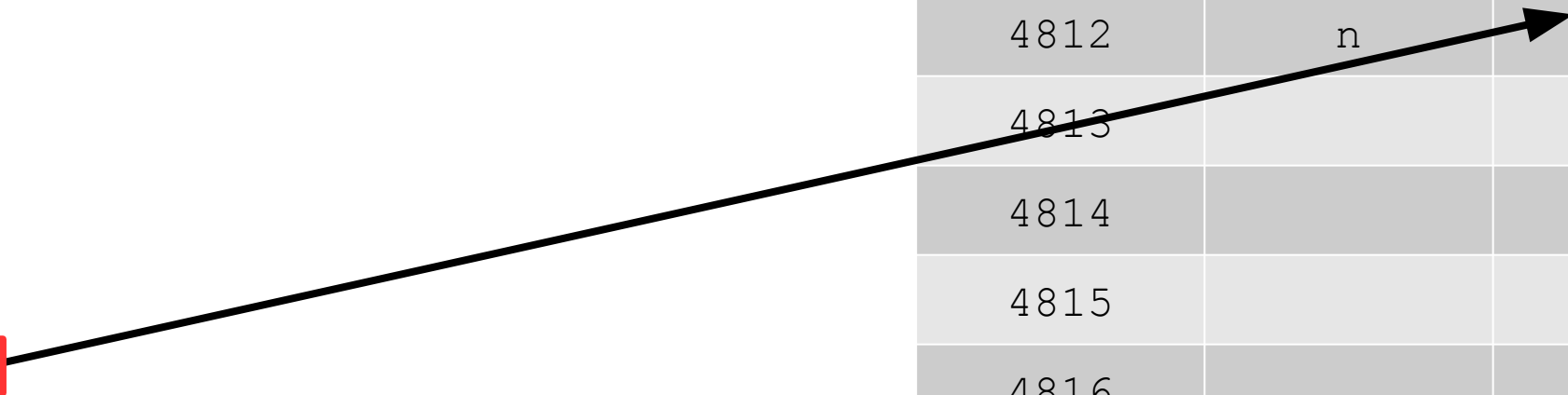
Escrevendo um programa em C

Exemplo

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

```
int main() {  
    float n;  
    n = 10.0;  
    n = n * 1.1;  
    n = n / 2;  
    int a = n;  
    printf("a = %d, n=%f", a, n);  
    getchar();  
    return 0;  
}
```

Endereço	Variável	Conteúdo
4812	n	5.5
4813		
4814		
4815		
4816		
4817		
4818		
4819		



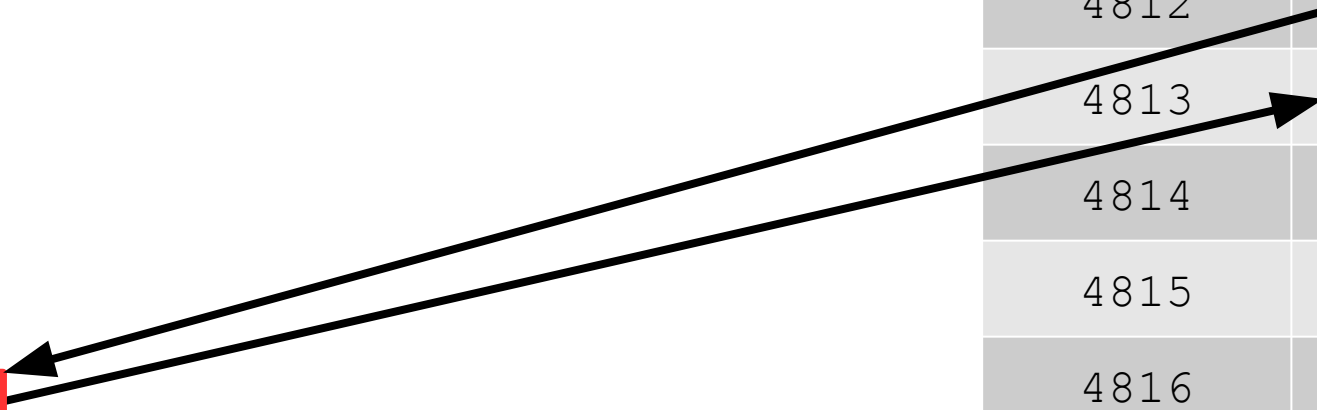
Escrevendo um programa em C

Exemplo

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

```
int main() {  
    float n;  
    n = 10.0;  
    n = n * 1.1;  
    n = n / 2;  
    int a = n;  
    printf("a = %d, n=%f", a, n);  
    getchar();  
    return 0;  
}
```

Endereço	Variável	Conteúdo
4812	n	5.5
4813		
4814		
4815		
4816		
4817		
4818		
4819		



Exercício

- Uma conta poupança foi aberta com um depósito de R\$500,00, com rendimentos 1% de juros ao mês. No segundo mês, R\$200,00 reais foram depositados nessa conta poupança. No terceiro mês, R\$50,00 reais foram retirados da conta. Quanto haverá nessa conta no quarto mês?

Perguntas?

- E-mail:
 - hector@dcc.ufmg.br
- Material da disciplina:
 - <https://pedroolmo.github.io/teaching/pdsI.html>
- Github:
 - <https://github.com/h3ct0r>



Héctor Azpúrua
h3ct0r