Revisão da Linguagem C

INF1005 -- Programação I -- 2016.1

Prof. Roberto Azevedo
razevedo@inf.puc-rio.br

exemplo de um programa em C

```c
#include <stdio.h>

int main (void) {
    float cels;
    float fahr;


    printf("Digite a temperatura em Celsius: ");

    scanf("%f", &cels);


    fahr = 1.8 * cels + 32;


    printf("Temperatura em Fahrenheit: %f", fahr);


    return 0;
}
```

exemplo de um programa em C

comentários são marcados entre /* e */

```c
/* Converte temperatura de Celsius em Fahrenheit */
#include <stdio.h> /* inclui biblioteca padrão */

int main (void) {
    float cels; /* temperatura em Celsius */
    float fahr; /* temperatura em Fahrenheit */

    /* exibe instrução para usuário */
    printf("Digite a temperatura em Celsius: ");
    /* lê da entrada padrao temperatura em Celsius */
    scanf("%f", &cels);

    /* calcula a conversão */
    fahr = 1.8 * cels + 32;

    /* exibe na tela o resultado */
    printf("Temperatura em Fahrenheit: %f", fahr);

    /* retorna 0 (indica execução normal do programa) */
    return 0;
}
```

exemplo de um programa em C

```c
/* Converte temperatura de Celsius em Fahrenheit */
#include <stdio.h> /* inclui biblioteca padrão */

int main (void) {
   float cels; /* temperatura em Celsius */
   float fahr; /* temperatura em Fahrenheit */

   /* exibe instrução para usuário */
   printf("Digite a temperatura em Celsius: ");
   /* lê da entrada padrao temperatura em Celsius */
   scanf("%f", &cels);

   /* calcula a conversão */
   fahr = 1.8 * cels + 32;

   /* exibe na tela o resultado */
   printf("Temperatura em Fahrenheit: %f", fahr);

   /* retorna 0 (indica execução normal do programa) */
   return 0;
}
```

função **main** indica o início do programa

exemplo de um programa em C

```c
/* Converte temperatura de Celsius em Fahrenheit */
#include <stdio.h> /* inclui biblioteca padrão */

int main (void) {
    float cels; /* temperatura em Celsius */
    float fahr; /* temperatura em Fahrenheit */

    /* exibe instrução para usuário */
    printf("Digite a temperatura em Celsius: ");
    /* lê da entrada padrao temperatura em Celsius */
    scanf("%f", &cels);

    /* calcula a conversão */
    fahr = 1.8 * cels + 32;

    /* exibe na tela o resultado */
    printf("Temperatura em Fahrenheit: %f", fahr);

    /* retorna 0 (indica execução normal do programa) */
    return 0;
}
```

cada bloco de instruções é marcado entre { e }

exemplo de um programa em C

```c
/* Converte temperatura de Celsius em Fahrenheit */
#include <stdio.h> /* inclui biblioteca padrão */

int main (void) {
   float cels; /* temperatura em Celsius */
   float fahr; /* temperatura em Fahrenheit */

   /* exibe instrução para usuário */
   printf("Digite a temperatura em Celsius: ");
   /* lê da entrada padrao temperatura em Celsius */
   scanf("%f", &cels);

   /* calcula a conversão */
   fahr = 1.8 * cels + 32;

   /* exibe na tela o resultado */
   printf("Temperatura em Fahrenheit: %f", fahr);

   /* retorna 0 (indica execução normal do programa) */
   return 0;
}
```

variáveis são declaradas com o seu **tipo** (número inteiro, real etc.)

exemplo de um programa em C

```c
/* Converte temperatura de Celsius em Fahrenheit */
#include <stdio.h> /* inclui biblioteca padrão */

int main (void) {
    float cels; /* temperatura em Celsius */
    float fahr; /* temperatura em Fahrenheit */

    /* exibe instrução para usuário */
    printf("Digite a temperatura em Celsius: ");
    /* lê da entrada padrao temperatura em Celsius */
    scanf("%f", &cels);

    /* calcula a conversão */
    fahr = 1.8 * cels + 32;

    /* exibe na tela o resultado */
    printf("Temperatura em Fahrenheit: %f", fahr);

    /* retorna 0 (indica execução normal do programa) */
    return 0;
}
```

printf é a função que exibe algo na tela

exemplo de um programa em C

```c
/* Converte temperatura de Celsius em Fahrenheit */
#include <stdio.h> /* inclui biblioteca padrão */

int main (void) {
    float cels; /* temperatura em Celsius */
    float fahr; /* temperatura em Fahrenheit */

    /* exibe instrução para usuário */
    printf("Digite a temperatura em Celsius: ");
    /* lê da entrada padrao temperatura em Celsius */
    scanf("%f", &cels);

    /* calcula a conversão */
    fahr = 1.8 * cels + 32;

    /* exibe na tela o resultado */
    printf("Temperatura em Fahrenheit: %f", fahr);

    /* retorna 0 (indica execução normal do programa) */
    return 0;
}
```

scanf é a função que lê valores da entrada padrão e os atribui à variáveis (neste caso apenas apenas um valor real, atribuído à variável *cels*)

exemplo de um programa em C

```c
/* Converte temperatura de Celsius em Fahrenheit */
#include <stdio.h> /* inclui biblioteca padrão */

int main (void) {
    float cels; /* temperatura em Celsius */
    float fahr; /* temperatura em Fahrenheit */

    /* exibe instrução para usuário */
    printf("Digite a temperatura em Celsius: ");
    /* lê da entrada padrao temperatura em Celsius */
    scanf("%f", &cels);

    /* calcula a conversão */
    fahr = 1.8 * cels + 32;

    /* exibe na tela o resultado */
    printf("Temperatura em Fahrenheit: %f", fahr);

    /* retorna 0 (indica execução normal do programa) */
    return 0;
}
```

comando de **atribuição** (variável *fahr* à direita do =)

exemplo de um programa em C

```c
/* Converte temperatura de Celsius em Fahrenheit */
#include <stdio.h> /* inclui biblioteca padrão */

int main (void) {
    float cels; /* temperatura em Celsius */
    float fahr; /* temperatura em Fahrenheit */

    /* exibe instrução para usuário */
    printf("Digite a temperatura em Celsius: ");
    /* lê da entrada padrao temperatura em Celsius */
    scanf("%f", &cels);

    /* calcula a conversão */
    fahr = 1.8 * cels + 32;

    /* exibe na tela o resultado */
    printf("Temperatura em Fahrenheit: %f", fahr);

    /* retorna 0 (indica execução normal do programa) */
    return 0;
}
```

retorno da main indica término do programa; por convenção 0 indica que não houve erro

tipos de dados

tipos de dados (cont.)

	tipos	bytes	valores
números inteiros	char	1	-128 a 127
	short int	2	-32.768 a 32.767
	long int	4	–2.147.483.648 a 2.147.483.647
	long long	8	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807
	int	(em geral = long int)	
números reais	float	4	~10^{-38} a 10^{38}
	double	8	~10^{-308} a 10^{308}

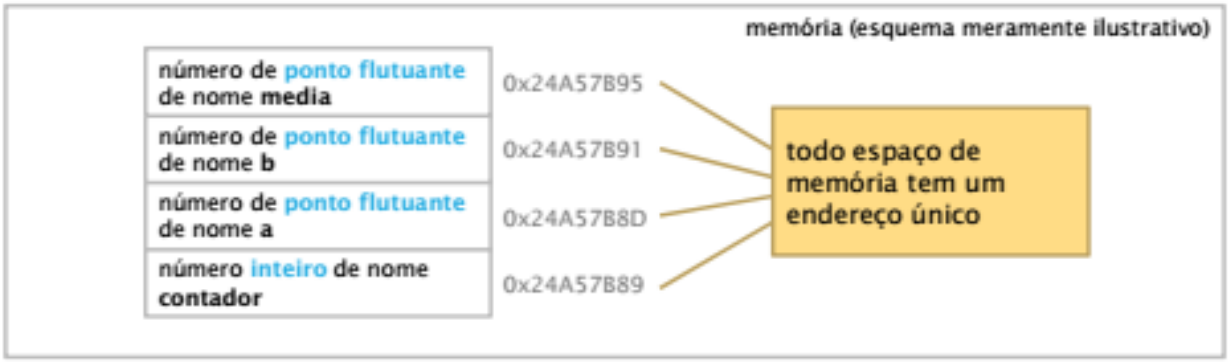
obs. também é possível aplicar o qualificador **unsigned** nos números inteiros

variáveis

o que é uma variável?

- O **nome** que se dá a um espaço reservado na memória, onde será possível armazenar um **valor** de um determinado **tipo**.
- Todas as variáveis devem ser explicitamente **declaradas**, antes de serem utilizadas:

```
float a, b;

int contador = 0;

float media = 0.0F;
```



como declarar uma variável?

declaração de variáveis

```
float tempC;
float tempF;
```

declaração de variáveis de um mesmo tipo (em uma só instrução)

```
float tempC, tempF;
int a, b;
```

inicialização de valor (atribuição após a declaração)

```
int a, b;
a = 5;
b = 10;
```

declaração com inicialização

```
int a = 5, b = 10;
char esc = '\\';
floar eps = 1.0e-5;
```

Toda variável deve ser inicializada antes de ser utilizada (ex.: exibida, incluída em uma expressão)

nomes de variáveis

- compostos por letras (A-Z, a-z, _) e dígitos (0-9).
- o primeiro caracter deve ser uma letra ('_' conta como letra).
- letras maiúsculas e minúsculas são diferentes.
- não podem ser palavras-chave da linguagem.

Nomes válidos

```
int contador = 0;

float a, b;

float media, _if,  _main;
```

Nomes inválidos

```
int 1contador = 0;

float a/2, b*2;

float 2media, if, main;
```

valores constantes

valores constantes

```
int a = 5;
int b;
b = a + 12;
```

(expressões constantes podem ser avaliadas em tempo de compilação)

exemplos

inteiros

```
13     -4
12345678L (long)
1234ul    (unsigned long)
31        (decimal)
031       (octal)
0x1f      (hexadecimal)
```

double

```
12.45
1245e-2
```

float

```
12.45F
```

char

```
'a'  'A'  '\t'
```

string (cadeia de caracteres)

```
"Rio de Janeiro"  "RJ"
"a"    "A"
```

sequências de escape

- Considerada um único caracter e, portanto, é valida como uma constante de caracter

\a	alarme	\\	contrabarra
\b	retrocesso	\?	ponto de interrogação
\f	alimentação de página	\’	apóstrofo
\n	nova linha	\”	aspas
\r	retorno de carro	\ooo	número octal
\t	tabulação horizontal	\xhhh	número hexadecimal
\v	tabulação vertical	\0	caracter com valor 0 ou NULL

conversão de tipo

implícita (automática, na avalição de uma expressão)

```
float a = 3; /* conversão para 3.0F */
```

explícita, através do operador *cast*

```
int x;
float y = 3.5F;
x = (int) y; /*descarta parte fracionária de y *;
```

operadores

operadores

aritméticos

+ - * / % (módulo)

relacionais

> >= < <= == !=

atribuição

= += -= *= /= %=

lógicos

&& (and)
|| (or)
! (not)

incremento e decremento

++ --

operadores aritméticos

\+ - * / % (módulo)

7 / 2 -> 3 (a parte fracionária é descartada)

7 / 2.0 -> 7.0 / 2.0 -> 3.5

7.0 / 2 -> 7.0 / 2.0 -> 3.5

(converte operandos para a maior precisão)

operadores de atribuição

=

a = 5;

y = x = 5;

operadores de atribuição compostos

i += 2 equivale a i = i + 2

i *= 2 equivale a i = i*2

i /= 2 equivale i = i / 2

...

var op= expr; equivale a var = var op (exp);

x *= y + 1; equivale ao quê?

```
x = x * (y+1);
```

operadores de incremento e decremento

após a variável

- "utiliza" o valor original, depois incrementa

x = 3

a = x++;

```
a = x; x = x + 1;
```

```
a = 3; x = 3 + 1;
```

antes da variável

- Primeiro incremente, depois "utiliza" o valor resultante

x = 3

a = ++x;

```
x = x + 1; a = x;
```

```
x = 3 + 1; a = 4;
```

precedência de operadores

tipo de operador	operador	associatividade
primários	`() [] . -> expr++ expr--`	esquerda para direita
unários	`* & + - ! ~ ++expr --expr (typecast) sizeof`	direita para esquerda
binários	`* / %`	esquerda para direita
	`+ -`	
	`>> <<`	
	`< > <= >=`	
	`== !=`	
	`&`	
	`^`	
	`|`	
	`&&`	
	`||`	
ternário	`? :`	direita para esquerda
atribuição	`= += -= *= /= %= >>= <<= &= ^= |=`	direita para esquerda

funções da biblioteca <stdio.h>

int printf (*formato, variáveis, …*): escreve na tela

```
int i = 3;
printf("%d\n", i);
```

```
3
```

alguns formatos:

%c char
%s : cadeia de caracteres
%d int
%u : unsigned int
%ld: long int
%f float
%lf: double
%e float (notação científica)
%le: double (notação científica)
%g float (formato mais curto: %f ou %e)
%lg: double (formato mais curto)
%x número em hexadecimal

tamanhos, posições, alinhamento:

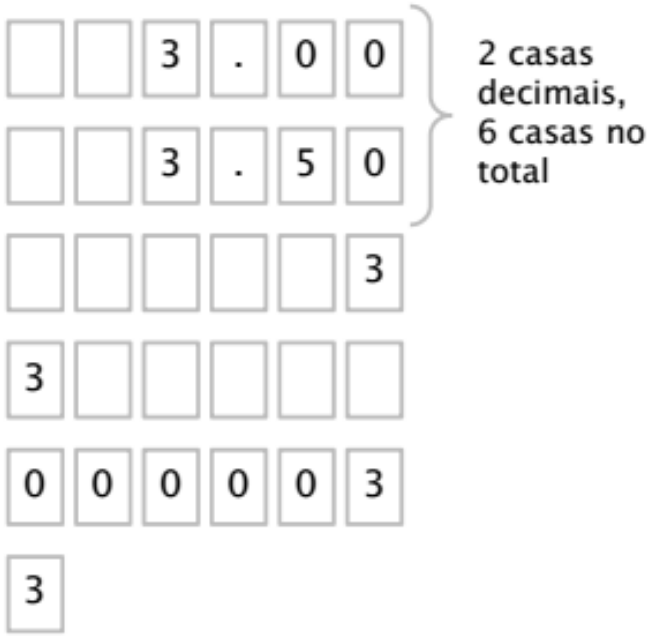
%4d inteiro em 4 posições
%-4d inteiro em 4 posições, alinhado à esquerda
%04d inteiro em 4 posições, preenchidas com zeros à esquerda do valor
%6.2f double (ou float) em 6 posições, com 2 casas decimais
%% sinal de porcentagem

alguns caracteres de escape:

\n caractere de nova linha
\" caractere "
\t caractere de tabulação
**** caractere \

printf - exemplos

```
float a = 3, b = 3.5;

int c = 3;

printf("%6.2f\n", a);

printf("%6.2f\n", b);

printf("%6d\n", c);

printf("%-6d\n", c);

printf("%06d\n", c);

printf("%d\n", c);
```



controle de fluxo - condicionais

condicionais (tomada de decisão) – if ... else

```
if ( expressão booleana ) /* se expressão for verdadeira */
{
    bloco de comandos 1  /
    ....
}
else                     /* senao */
{
    bloco de comandos 2  /
    ....
}

próximo comando ....     /* prossegue para o próximo comando após o if */
```

verdadeiro: expressao != 0

falso: expressao == 0

exemplo

```
if ( media >= 6.0 )              /* se expressão for verdadeira */
{
    printf("Aprovado");          /* ... Executa o bloco de comandos 1, */
}
else                             /* senao */
{
    printf("Em prova final\n");  /* ... Executa o bloco de comandos 2, */
}
printf ("\nfim"); ..           /* prossegue para o próximo comando após o if */
```

condicionais (tomada de decisão) – if

expressão booleana

- é uma expressão que, quando avaliada, resulta no valor **falso** ou **verdadeiro**.
- a linguagem C **não** tem um tipo de dado específico para armazenar valores *booleanos*:
  - Em C, o valor booleano é representado por um valor inteiro:
    - 0 significa falso e qualquer outro valor diferente de zero significa verdadeiro.
    - Em geral, usa-se 1 para representar o valor verdadeiro, e qualquer expressão booleana que resulta em verdadeiro resulta no valor 1.

operadores relacionais

> < <= >= == !=

numa expressão booleana, a comparação de dois operandos resulta em

0, caso a expressão seja **falsa**, ou

1, caso a expressão seja **verdadeira**

Exemplo:

```
int a=10, b=10, c=5;
a > b     resulta em 0 (falso)
b > c     resulta em 1 (verdadeiro)
a == b    resulta em 1 (verdadeiro)
b != c    resulta em 1 (verdadeiro)
```

operadores lógicos

combinam expressões ou valores booleanos

conjunção (and)	(operador binário)
&&	a && b
disjunção (or)	(operador binário)
||	a || b
negação (not)	(operador unário)
!	!a

operadores lógicos - conjunção

&& (and) || (or) ! (not)

true && true resulta em true

true && false resulta em false

false && true resulta em false

false && false resulta em false

exemplos

(1 > 0) && (2 > 0) é avaliada como 1 && 1, e resulta em 1 (verdadeiro)

(1 > 0) && (2 < 0) é avaliada como 1 && 0, e resulta em 0 (falso)

(2 < 0) && (1 > 0) resulta em 0 (false)

(1 < 0) && (2 < 0) resulta em 0 (false)

operadores lógicos – disjunção

&& (and) **|| (or)** ! (not)

true || true resulta em true

true || false resulta em true

false || true resulta em true

false || false resulta em false

exemplos

(1 > 0) || (2 > 0) resulta em 1 (verdadeiro)

(1 > 0) || (2 < 0) resulta em 1 (verdadeiro)

(2 < 0) || (1 > 0) é avaliada como 0 || 1, e resulta em 1 (verdadeiro)

(1 < 0) || (2 < 0) é avaliada como 0 || 0, e resulta em 0 (falso)

operadores lógicos – negação

&& (and) || (or) **! (not)**

!true resulta em false

!false resulta em true

exemplos

!(1 > 0) é avaliada como !1, e resulta em 0 (falso)

!(1 < 0) é avaliada como !0, e resulta em 1 (verdadeiro)

condicionais (tomada de decisão) – encadeamento if ... else if ... else

```
if ( expressao1 )          /* se expressão1 for verdadeira */
{
    bloco de comandos 1    /* ... Executa o bloco de comandos 1, */
}
else if ( expressao2 )     /* senao  se expresão2 for verdadeira */
{
    bloco de comandos 2    /* ... Executa o bloco de comandos 2, */
}
else if ( expressao3 )     /* senao  se expresao3 for verdadeira */
{
    bloco de comandos 3    /* ... Executa o bloco de comandos 3, */
}
else                       /* senao  */
{
    bloco de comandos n    /*executado quando todas as expressões são falsas*/
}

próximo comando ....      /* prossegue para o próximo comando após o if */
```

controle de fluxo – repetições

dúvidas?

```
while ( expressao )
{
    bloco de comandos
}
```

1) se *expressao* for verdadeira, executa bloco de comandos
2) repete a partir de (1)

```
do
{
    bloco de comandos
} while ( expressao );
```

1) executa bloco de comandos
2) se *expressao* for verdadeira, repete a partir de (1)

```
for (expr_inicial; expr_teste_laço; expr_atualização)
{
    bloco de comandos
}
```

1) avalia *expr_inicial*
2) se *expr_teste_laço* for verdadeira:
   2.1) executa bloco de comandos
   2.2) avalia *expr_atualização*
   2.3) repete a partir de (2)

```
expr_inicial;
while (expr_teste_laço)
{
    bloco de comandos
    expr_atualização
}
```

PUC RIO

DEPARTAMENTO DE INFORMÁTICA PUC-RIO

construção for

```
for ( expr_inicial; expr_teste_laço; expr_atualização)
{
    bloco de comandos
}
```

1) avalia `expr_inicial`
2) se `expr_teste_laço` for verdadeira:
   2.1) executa bloco de comandos
   2.2) avalia `expr_atualização`
   2.3) repete a partir de (2)

Construção equivalente usando while:

```
expr_inicial;
while (expr_teste_laço)
{
    bloco de comandos
    expr_atualização
}
```

laços – interrupções com break e continue

break

- termina o laço e transfere a execução para a instrução imediatamente depois do laço

continue

- faz com que o laço pule o restante do corpo e imediatamente re-teste a condição antes de iterar novamente

laços – interrupções com break e continue

```
#include <stdio.h>
int main (void)
{
  int i;
  for ( i = 0; i < 10; i ++) {
    if (i == 5)
      break;
    printf("%d ", i);
  }
  printf ("fim.\n");
  return 0;
}
```

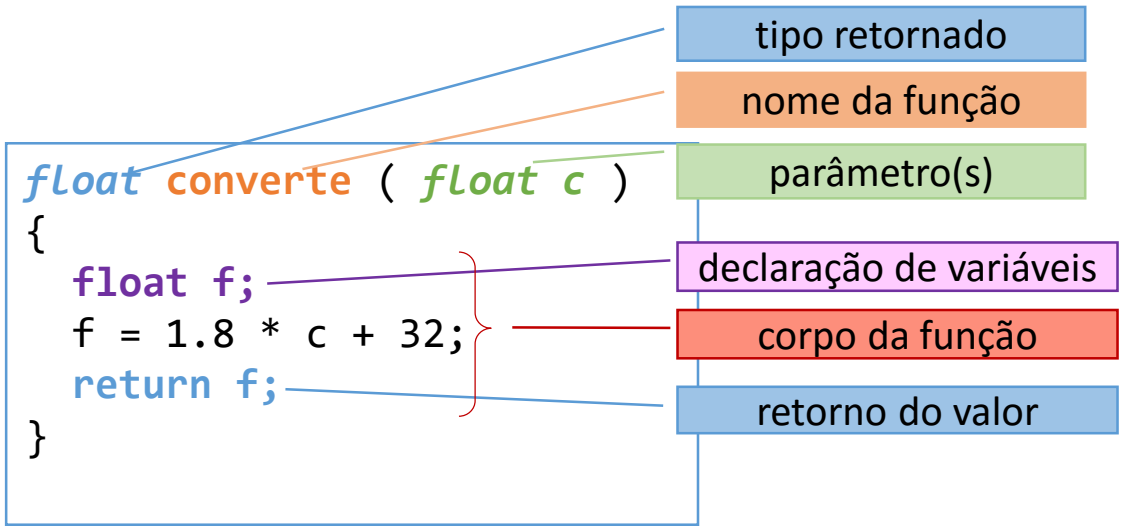
```
#include <stdio.h>
int main (void)
{
  int i;
  for ( i = 0; i < 10; i ++) {
    if (i == 5)
      continue;
    printf("%d ", i);
  }
  printf ("fim.\n");
  return 0;
}
```

Qual é a saída de cada um desses programas?

funções

como definir uma função

```
tipo_retornado nome_da_função ( lista de parâmetros ... )
{
    corpo_da_função
}
```



Sintaxe
(forma da função)

semântica
(funcionalidade ou comportamento; o que a função faz).