Linguagem de Programação C

Funções e Procedimentos

Cristiano Lehrer

http://www.ybadoo.com.br/

Introdução (1/5)

- Embora sem saber ainda como escrever uma função, já utilizamos várias em nossos exemplos:
  - `printf()`, `scanf()`, `getchar()`, `putchar()`, ...
- Escreva um programa que coloque na tela a seguinte saída:

```
**********************
Números entre 1 e 5
**********************
1
2
3
4
5
**********************
```

Introdução (2/5)

```c
#include <stdio.h>
int main() {

  int i;

  // Imprimir *******
  for(i = 0; i < 20; i++)
    putchar('*');
  putchar('\n');

  puts("Números de 1 a 5");

  // Imprimir *******
  for(i = 0; i < 20; i++)
    putchar('*');
  putchar('\n');

  // Imprimir os números
  for(i = 1; i < 6; i++)
    printf("%d\n", i);

  // Imprimir *******
  for(i = 0; i < 20; i++)
    putchar('*');
  putchar('\n');

  return 0;
}
```

Introdução (3/5)

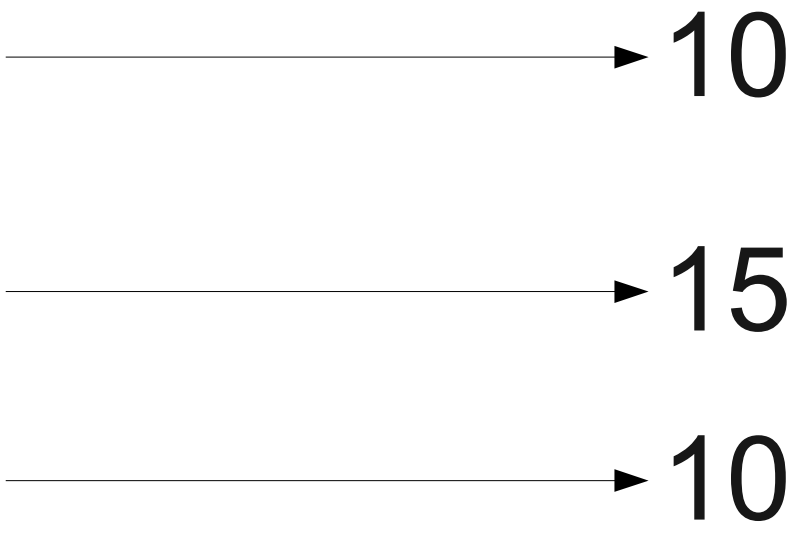
```c
#include <stdio.h>

void linha(){
  int i;
  for(i = 0; i < 20; i++)
    putchar('*');
  putchar('\n');
}

int main(){
  int i;
  linha();
  puts("Números de 1 a 5");
  linha();
  // Imprimir os números
  for(i = 1; i < 6; i++)
    printf("%d\n", i);

  linha();

  return 0;
}
```

Introdução (4/5)

- Um programa em C tem que possuir sempre a função `main()` escrita no seu código, independentemente do número e da variedade de funções que o programa contenha.
- As variáveis declaradas dentro de um bloco são locais a esse bloco, não sendo conhecidas fora dele.

Introdução (5/5)

```c
#include <stdio.h>

int main()
{
  int i = 10;

  printf("%d\n", i);
  {
    int i = 15;

    printf("%d\n", i);
  }

  printf("%d\n", i);

  return 0;
}
```



Características de uma Função (1/2)

- Cada função tem que ter um nome único, o qual serve para a sua invocação em algum lugar no programa a que pertence.
- Uma função pode ser invocada a partir de outras funções.
- Uma função (como o seu nome indica) deve realizar uma única tarefa bem definida.
- Uma função deve comportar-se como uma caixa preta:
  - Não interessa como funciona, o que interessa é que o resultado final seja o esperado, sem efeitos colaterais.

Características de uma Função (2/2)

- O código de uma função deve ser o mais independente possível do resto do programa, e deve ser tão genérico quanto possível, para poder ser reutilizado em outros projetos.
- Uma função pode receber parâmetros que alterem o seu comportamento de forma a adaptar-se facilmente a situações distintas.
- Uma função pode retornar, para a entidade que a invocou, um valor como resultado do seu trabalho.

Nome de uma Função (1/3)

- A escolha do nome de uma função obedece às regras para a designação de variáveis:
  - O nome de uma variável pode ser constituído por letras do alfabeto (minúsculas ou maiúsculas), dígitos (0..9) e ainda pelo caractere *underscore* (_).
  - O primeiro caractere não pode ser um dígito. Terá que ser uma letra ou o caractere *underscore*. No entanto, é desaconselhável a utilização deste último como primeira letra identificadora de uma variável.
  - Maiúsculas e minúsculas representam caracteres diferentes, logo representam variáveis distintas.
  - Uma variável não pode ter por nome uma palavra reservada da própria Linguagem C. Assim, não pode ter uma variável denominada `float`, `if` ou `for`, uma vez que essas palavras são instruções ou tipos da própria linguagem.

Nome de uma Função (2/3)

- O nome de uma função deve ser único (não pode ser igual ao nome de outra função ou de uma variável).
- O nome de uma função deve especificar aquilo que a função na realidade faz, e deve ser de fácil leitura e interpretação.

Nome de uma Função (3/3)

- Cuidados a seguir para nomes de funções e variáveis:
  - O nome deve ser descritivo daquilo que a variável armazena ou a função executa.
  - O nome não deve ser todo escrito em maiúsculas, pois identificadores totalmente escritos em maiúsculas são tradicionalmente utilizados pelos programadores de C para referenciar constantes.
  - Caso o nome use mais do que uma palavra, utilize o caractere *underscore* ou a diferença entre minúsculas e maiúsculas para as separar, facilitando assim a leitura.
  - Não é aconselhável a utilização de caracteres acentuados no nome, pois a grande maioria dos compiladores não os aceita como caracteres admissíveis.

Como Funciona uma Função

- O código de uma função só é executado quando está é invocada em alguma parte do programa a que está de algum modo ligada.
- Sempre que uma função é invocada, o programa que a invoca é suspenso temporariamente. Em seguida, são executadas as instruções presentes no corpo da função. Uma vez terminada a função, o controle de execução do programa volta ao local em que esta foi invocada.
- O programa que invoca uma função pode enviar argumentos, que são recebidos pela função. Estes são recebidos e armazenados em variáveis locais, que são automaticamente iniciadas com os valores enviados. A essas variáveis dá-se o nome de parâmetros.
- Depois de terminar o seu funcionamento, uma função pode devolver um valor para o programa que a invocou.

Exemplo (1/6)

- Escreva um programa em C, que escreva na tela a seguinte saída:

```
  ***
 *****
*******
 *****
  ***
```

Exemplo (2/6)

```c
#include <stdio.h>

void linha3x(){
  int i;
  for(i = 0; i < 3; i++)
    putchar('*');
  putchar('\n');
}

void linha5x(){
  int i;
  for(i = 0; i < 5; i++)
    putchar('*');
  putchar('\n');
}
```

```c
void linha7x(){
  int i;
  for(i = 0; i < 7; i++)
    putchar('*');
  putchar('\n');
}

int main(){
  linha3x();
  linha5x();
  linha7x();
  linha5x();
  linha3x();

  return 0;
}
```

Exemplo (3/6)

```c
#include <stdio.h>

void linha(int num){
  int i;
  for(i = 0; i < num; i++)
    putchar('*');
  putchar('\n');
}
```

```c
int main(){
  linha(3);
  linha(5);
  linha(7);
  linha(5);
  linha(3);

  return 0;
}
```

Qualquer tipo de dado da linguagem pode ser enviado como parâmetro para uma função, mesmo os tipos de dados que venham a ser definidos pelo programador.

Exemplo (4/6)

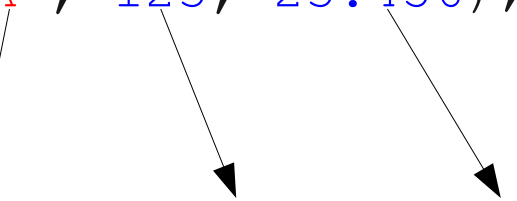
- Um parâmetro não é nada mais do que uma variável local à função a que pertence.
- Um parâmetro é automaticamente iniciado com o valor enviado pelo programa invocador.

```
/* Exemplo correto */
funcao(int x, int y, int k, int xpto)

/* Exemplo incorreto */
funcao(int x, y, k, xpto)
```

Exemplo (5/6)

- O número de parâmetros enviados para uma função deve ser igual ao número de parâmetros existentes no cabeçalho da função.
- O tipo dos parâmetros deve igualmente corresponder, parâmetro por parâmetro.

```
int main()
{
   ...
   funcao('A', 123, 23.456);
   ...
}

funcao(char ch, int n, float x)
{
   ...
}
```



Exemplo (6/6)

```c
#include <stdio.h>

void linha(int num, char ch){
  int i;
  for(i = 0; i < num; i++)
    putchar(ch);
  putchar('\n');
}
```

```c
int main(){
  linha(3, '+');
  linha(5, '+');
  linha(7, '-');
  linha(5, '*');
  linha(3, '*');

  return 0;
}
```

```
+++
+++++
-------
*****
***
```

Corpo da Função (1/2)

- O cabeçalho de uma função nunca deve ser seguido de ponto-e-vírgula (;).

```c
/* errado */
void linha(int num);
{
  int i;
  for(i = 0; i < num; i++)
    putchar('*');
  putchar('\n');
}
```

```c
/* correto */
void linha(int num)
{
  int i;
  for(i = 0; i < num; i++)
    putchar('*');
  putchar('\n');
}
```

Corpo da Função (2/2)

- Dentro do corpo de uma função pode-se escrever qualquer instrução ou conjuntos de instruções da linguagem C.
- Em C não se pode definir funções dentro de funções.

```c
/* errado */
void linha(int num){

  void escrever(char ch){
    putchar(ch);
  }

  int i;
  for(i = 0; i < num; i++)
    escrever('*');
  putchar('\n');
}
```

```c
/* correto */
void escrever(char ch){
  putchar(ch);
}

void linha(int num){
  int i;
  for(i = 0; i < num; i++)
    escrever('*');
  putchar('\n');
}
```

Funções que Retornam um Valor (1/4)

- A instrução `return` permite terminar a execução de uma função e voltar ao programa que a invocou.
- A execução da instrução `return` na função `main` faz com que o programa termine.

```
int main()
{
  printf("Hello");

  return 0;

  printf(" World!\n");
}
```

Funções que Retornam um Valor (2/4)

- A devolução de um resultado é feita através da instrução `return`, seguida do valor a ser devolvido.
- Após a instrução `return` pode ser colocada qualquer expressão válida em C.

```
int soma(int x, int y)
{
  int res;

  res = x + y;

  return res;
}
```

Funções que Retornam um Valor (3/4)

- Uma função pode ser invocada dentro de outra função.
- O resultado é o mesmo que se obteria se, em vez da chamada à função, aí estivesse o resultado devolvido por esta.

```
int soma(int x, int y){
  return (x + y);
}

int dobro(int x){
  return (x * 2);
}
```

```
int main(){
  ...
  printf("%d\n", dobro(2));
  printf("%d\n", soma(dobro(2), 3));
  printf("%d\n", dobro(soma(dobro(2), 3)));
  ...
}
```

Funções que Retornam um Valor (4/4)

- Uma função pode conter várias instruções `return`.
- No entanto, apenas uma instrução `return` é executada na função.

```
int max(int n1, int n2)
{
  if(n1 > n2)
    return n1;
  else
    return n2;
}
```

Funções e Procedimentos

- Uma função tem sempre um tipo e um valor de retorno associados, enquanto que um procedimento não devolve qualquer valor:
  - Função:
    - `int max(int n1, int n2)`
  - Procedimento (instruções equivalentes):
    - `void linha()`
    - `void linha(void)`
  - Observação (instruções equivalentes):
    - `linha()`
    - `int linha()`

Onde Colocar as Funções (1/2)

- As funções podem ser colocadas em qualquer lugar dentro de um arquivo C.
- Observação:
  - Se forem implementadas após o seu uso, o compilador pode acusar um erro.

```
...
main(){
  linha();
  printf("Hello World!\n");
  linha();
}
...
void linha(){
  for(int i = 0; i < 20; i++)
    putchar('*');
  putchar('\n');
}
...
```

Onde Colocar as Funções (2/2)

- O protótipo de uma função corresponde ao seu cabeçalho seguido de um ponto-e-vírgula (;).
- Este identifica toda a estrutura da função (nome, parâmetros e tipo de retorno).

```
...
void linha();
...
main(){
  linha();
  printf("Hello World!\n");
  linha();
}
...
void linha(){
  for(int i = 0; i < 20; i++)
    putchar('*');
  putchar('\n');
}
...
```