ANSI

C

Introdução à Programação

Eduardo Silva Lira

XLVIII Programa de Verão do IME-USP

São Paulo - SP, Jan 2019

INSTITUTO DE MATEMATICA E ESTATÍSTICA
UNIVERSIDADE DE SÃO PAULO

USP

Funções: cabeçalho

Questão:

```c
/*Exemplo de função para somar dois valores*/
#include<stdio.h>

int main(){
  int a = 5, b = 7;
  printf("soma: %d\n", somar(a, b));
  return 0;
}

int somar(int a, int b){
  int resultado;
  resultado = a + b;
  return resultado ;
}
```

Conseguiremos compilar e rodar este programa?

Funções: cabeçalho

- Definir a estrutura básica da função
  - Tipo de retorno
  - Identificador
  - Parâmetros
- Implementação posterior
- Aprimora o funcionamento do compilador

Funções: cabeçalho

Exemplo:

```c
/*Exemplo de função para somar dois valores*/
#include<stdio.h>

int somar(int, int);

int main(){
  int a = 5, b = 7;
  printf("soma: %d\n", somar(a, b));
  return 0;
}

int somar(int a, int b){
  int resultado;
  resultado = a + b;
  return resultado ;
}
```

Funções: cabeçalho

Exemplo:

```c
/*Exemplo de função para somar dois valores*/
#include<stdio.h>

int somar(int, int);

int main(){
  int a = 5, b = 7;
  printf("soma: %d\n", somar(a, b));
  return 0;
}

int somar(int a, int b){
  int resultado;
  resultado = a + b;
  return resultado ;
}
```

Aqui nós colocamos o **cabeçalho** da função

Aqui nós **implementamos** a função

Funções: cabeçalho

- Como se diz cabeçalho em inglês?
- O que quer dizer o "`.h`" no `#include<stdio.h>`?
- Onde isto irá nos levar?

Funções: cabeçalho

- Como se diz cabeçalho em inglês?
- O que quer dizer o "`.h`" no `#include<stdio.h>`?
- Onde isto irá nos levar?
  - Lembram do "*Escreva uma vez, utilize onde quiser*"?

Funções: cabeçalho

- Como se diz cabeçalho em inglês?
- O que quer dizer o "`.h`" no `#include<stdio.h>`?
- Onde isto irá nos levar?
  - Lembram do "*Escreva uma vez, utilize onde quiser*"?

Iremos criar nossas próprias **bibliotecas**

Como criar minhas bibliotecas?

Crie um arquivo de **header**, com os **protótipos** de funções. Salve como **minhalib.h**

```
#ifndef _MINHALIB_H_
#define _MINHALIB_H_

/*Cabecalho das minhas funcoes*/
int somar(int, int);
double dobro(double);

#endif
```

Como criar minhas bibliotecas?

- Crie um arquivo **minhalib.c**
- Inclua o arquivo de **header**
  - **#include “minhalib.h”**
  - Implemente **todas** as funções
    - Programe!

Como criar minhas bibliotecas?

Arquivo **minhalib.c**:

```c
#include "minhalib.h"

int somar(int a, int b){
  return a + b;
}

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

Como criar minhas bibliotecas?

- Preparar a biblioteca para ser utilizada
  - Geração de código objeto

gcc -Wall -ansi -pedantic-errors -c minhalib.c -o minhalib.o

  - Agora utilize a biblioteca em seus programas!

Como criar minhas bibliotecas?

Arquivo **expLib.c**: utilizar a biblioteca criada

```c
#include<stdio.h>
#include "minhalib.h"

int main(){
   int a = 10, b = 70;
   double x = 5.7, y = 7.0;
   printf("Soma de a e b: %d\n", somar(a, b));
   printf("Dobro de x: %f\n", dobro(x));
   printf("Dobro de y: %f\n", dobro(y));
   return 0;
}
```

Como criar minhas bibliotecas?

- Compile seu programa
  - Observe que a biblioteca vai no comando!

gcc -Wall -ansi -pedantic-errors expLib.c -o expLib minhalib.o

Como criar minhas bibliotecas?

- Compile seu programa
  - Observe que a biblioteca vai no comando!

gcc -Wall -ansi -pedantic-errors

expLib.c -o expLib minhalib.o

Como criar minhas bibliotecas?

- Compile seu programa
  - Observe que a biblioteca vai no comando!

gcc -Wall -ansi -pedantic-errors expLib.c -o expLib minhalib.c

Tente também adicionar direto o arquivo .c

Dicas! Bibliotecas e funções

- stdio.h
  - remove
  - rename
  - fread
  - fwrite

Dicas! Bibliotecas e funções

- ctype.h
  - isalpha(c)
  - isalphanum(c)
  - isupper(c)
  - islower(c)
  - iscntrl(c)
  - tolower(c)
  - toupper(c)

Dicas! Bibliotecas e funções

- string.h
  - char *strcpy(s, cp)
  - int strcmp(cs, cp)
  - size_t strlen(cs)
  - char *strncat(s,ct,n)

Dicas! Bibliotecas e funções

- math.h
  - sin(x)
  - cos(x)
  - tan(x)
  - log10(x)
  - pow(x, y)
  - sqrt(x)
  - floor(x)

Dicas! Bibliotecas e funções

- time.h
  - time_t time(time_t *tp)
  - double difftime(time_t time2, time_t time1)

Dicas! Bibliotecas e funções

- stdlib.h
  - int atoi(const char *s)
  - long atol(const char *s)
  - int system(const char *s)
  - int abs(int n)
  - int rand(void)
  - void srand(unsigned int seed)

Dicas! Bibliotecas e funções

- limits.h e float.h
  - Constantes com os maiores e menores valores armazenáveis
    - CHAR_MAX
    - LONG_MAX
    - SHRT_MIN
    - DBL_MIN

Dica!

- Ponteiros podem ou devem ser inicializados?!
  - **NULL**

Alocação dinâmica de memória

- É possível alocar a quantidade desejada de memória, no momento desejado?
  - Não sei quantos alunos uma escola tem
    - O meu sistema roda em escolas diferentes

Alocação dinâmica de memória

- Utilizar ponteiro
  - O sistema nos disponibiliza um bloco
  - Apontamos para este bloco

Alocação dinâmica de memória

- Funções úteis
  - malloc
  - realloc
  - calloc
  - free

Alocação dinâmica de memória

- Funções úteis
  - malloc
  - realloc
  - calloc
  - free

- Onde estas funções estão?

  #include <stdlib.h>

Alocação dinâmica de memória

- Funções úteis
  - **malloc**
    - Aloca **n** bytes
  - **realloc**
    - Altera quantidade de armazenamento
    - Realoca se necessário
  - **calloc**
    - Aloca **n** posições de tamanho **size**
    - Inicializa-os para **zero**
  - **free**
    - Devolte ao sistema um bloco

Alocação dinâmica de memória

- Funções úteis
  - **malloc**
    - Aloca **n** bytes
  - **realloc**
    - Altera quantidade de armazenamento
    - Realoca se necessário
  - **calloc**
    - Aloca **n** posições de tam[illegible] **size**
    - Inicializa-os para **zero**
  - **free**
    - Devolte ao sistema um bloco

Alocação dinâmica de memória

- Função **malloc:** Aloca **size** bytes
  - Sintaxe
    - Retorna um ponteiro
    - Retorna **NULL** se não alocou
    - Recebe **size** em bytes

```
void *malloc(size_t size);
```

Alocação dinâmica de memória

- Função **calloc**: Aloca **n** objetos de tamanho **size** bytes. Inicializa com zero.
  - Sintaxe
    - Retorna um ponteiro
    - Retorna **NULL** se não alocou
    - Recebe **size** em bytes
    - Recebe também **nobj**

```
void *calloc(size_t nobj, size_t size);
```

Alocação dinâmica de memória

- Função **realloc**: realocar o objeto para o qual **p** aponta.
  - Sintaxe
    - Retorna um ponteiro
    - Retorna **NULL** se não encontrou
    - Recebe **size** em bytes (novo tamanho)
    - Recebe o ponteiro **p**

```
void *realloc(void *p, size_t size);
```

Liberação de memória

- Espaços não mais ocupados são liberados automaticamente em C?

Liberação de memória

- Espaços não mais ocupados são liberados automaticamente em C?
  - Defina não mais ocupados!
    - Alocação **estática**
      - Sim
    - Alocação **dinâmica**
      - Não

Liberação de memória

- Espaços não mais ocupados são liberados automaticamente em C?
  - A solução:
    - Utilizar a função **free**

Alocação dinâmica - Exercícios

- Faça um programa para
  - Ler uma quantidade **n**
  - Alocar **n** espaços em memória para guardar números inteiros positivos
  - Ao digitar um valor inválido, exibir todos os valores, na ordem digitada

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

int main () {
   char src[40];
   char dest[100];

   memset(dest, '\0', sizeof(dest));
   strcpy(src, “Curso de Verão IME“);
   strcpy(dest, src);

   printf(“String final: %s\n", dest);

   return(0);
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

int main () {
   char src[50], dest[50];

   strcpy(src,  “Este é a origem“);
   strcpy(dest, “Este é o destino“);

   strncat(dest, src, 15);

   printf(“String final: l%sl", dest);

   return(0);
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

int main () {
   time_t seconds;

   seconds = time(NULL);
   printf("Hours since January 1, 1970 = %ld\n", seconds/3600);

   return(0);
}
```

```
#include <stdlib.h>
#include <time.h>
#include <string.h>

int main () {
   int i, n;
   time_t t;

   n = 5;

   //srand(2);
   srand((unsigned) time(&t));

   for( i = 0 ; i < n ; i++ ) {
      printf("%d\n", rand() % 50);
   }

   return(0);
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

int main(){
  int *p = NULL;
  int a = 2;
  p = &a;
  *p += 2;
  printf("O valor do ponteiro é: %p\n", p);
  printf("O valor da variável apontada pelo ponteiro”);
  printf(“ é: %d\n", *p);
  return(0);
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

int main () {
   char *str;
   str = (char *) malloc(15);
   strcpy(str, "tutorialspoint");
   printf("String = %s,  Address = %p\n", str, str);
   str = (char *) realloc(str, 25);
   strcat(str, ".com");
   printf("String = %s,  Address = %p\n", str, str);

   free(str);
   return(0);
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

int main () {
   int i, n, *a;
   printf("Número de elementos:");
   scanf("%d",&n);
   a = (int*)calloc(n, sizeof(int));
   printf("Entre os %d números:\n",n);
   for( i=0 ; i < n ; i++ ) scanf("%d",&a[i]);
   printf("Os números são: ");
   for( i=0 ; i < n ; i++ ) printf("%d ",a[i]);

   free( a );
   return(0);
}
```

- Para saber mais sobre as opções do compilador gcc:
  - https://gcc.gnu.org/onlinedocs/gcc/Option-Index.html

- Mais sobre bibliotecas:
  - https://www.tutorialspoint.com/c_standard_library/index.htm

- Alguns fóruns ativos para tirar dúvidas
  - http://clubedaprogramacao.com/forum/index.php
  - https://pt.stackoverflow.com/

Dúvidas?