Linguagem de Programação

Strings

Funções

String

- caractere: letra, numeral, pontuação, símbolo
- String: termo utilizado para identificar um conjunto de caracteres

String

- Declaração e atribuição de caractere em C:
- char opção = ‘A’;

String

- Linguagem C:
- Conjunto de char terminado por (‘\0’)
- char str [10];

String

- Definição:
- Char nome[5];
- Nome: nome da variável
- [5]: numero de posições de 0 a 5
- Ultima posição é ocupada pelo ‘\0’

String - Exemplo

```
#include <stdio.h>
void main(){
        char nome [3];
        nome[0] = 'M';
        nome[1] = 'A';
        nome[2] = 'E';
        nome[3] = '\0';
        printf("O nome é %s\n", nome);
        printf("A terceira letra da string eh %c\n",
nome[2]);
}
```

String

- Leitura de string(teclado):
- Função gets()
- Lê string até o primeiro enter
- Função scanf()
- Lê string até o primeiro espaço em branco

String – exemplo de funções

- Necessário biblioteca string.h
- **strcpy (str1, str2)**

Copia str2 em str1

- **strcat(str1, str2)**

Concatena str2 ao final de str1

- **strlen(str1)**

Retorna o tamanho de str1

- **strcmp(str1,str2)**

Retorna 0 se str1 e str2 são iguais; < 0 se str1<str2; > 0 se str1>str2

Modularização

Funções e procedimentos

Modularização

- Dividir o problema em partes denominadas módulos
- Cada módulo:
  - Conjunto de tarefas implementadas para resolver o problema de maneira eficiente
- Algoritmo principal:
  - Gerencia as tarefas
  - Chama ou aciona outros módulos
- Sub-rotinas

Sub-rotinas

- Subprogramas
- Funções
- Procedimentos
- Blocos de instruções que realizam tarefas específicas

Sub-rotinas

- Funções
  - Retornam valor
- Procedimentos
  - Sem retorno de valor

Vantagens

- Reaproveitamento de código
- Evita Repetição trecho de código
- Permite alteração de um trecho de código de forma mais rápida
- Evita blocos do programa grandes demais
- Facilita a leitura do programa-fonte
- Separa o programa em partes(blocos) que possam ser logicamente compreendidos isoladamente

C

Sub-rotinas

Funções pré definidas

- printf(“Olá mundo”)
- scanf(“%d”, &numero)
- getch(letra)
- strcpy(string,string)
- strlen(“Oi”)

Funções definidas pelo programador

```
#include <stdio.h>
void soma( ){
	int resultado, num1, num2;
	printf("\n digite o primeiro numero: ");
	scanf("%d", &num1);
	printf("\n digite o segundo numero: ");
	scanf("%d", &num2);
	resultado = num1 + num2;
	printf("\n a soma é: %d", resultado);
}
int main(int argc, char *argv[]) {
	soma();
	return 0;
}
```

Sub-rotinas : funções e procedimentos

- Pode receber diversos valores (parâmetros)
- Pode retornar valor
- O tipo de retorno:
  - obrigatório
  - determina se é função ou procedimento

Sub-rotinas : funções e procedimentos

- Parâmetros:
- Formais: utilizados na definição da função. São variáveis.
- Reais: usados na chamada da função. Pode, ser tanto variáveis quanto valores.

Sub-rotinas : funções e procedimentos

- Exemplo de utilização de parâmetros:

```
int soma(int num1, int num2){
        return num1 + num2;
}


int main(int argc, char *argv[]) {
        int res;
        res = soma(2,3);
        return 0;
}
```

Sub-rotina sem passagem de parâmetro e sem retorno

- Não recebe informação no momento da chamada
- Não repassa valor

Sub-rotina sem passagem de parâmetro e sem retorno

```
#include <stdio.h>
void soma( ){
	int resultado, num1, num2;
	printf("\n digite o primeiro numero: ");
	scanf("%d", &num1);
	printf("\n digite o segundo numero: ");
	scanf("%d", &num2);
	resultado = num1 + num2;
	printf("\n a soma é: %d", resultado);
}
int main(int argc, char *argv[]) {
	soma();
	return 0;
}
```

Sub-rotina sem passagem de parâmetro e com retorno

- Não recebe informação no momento da chamada
- Retorna valor

Sub-rotina sem passagem de parâmetro e com retorno

```
#include <stdio.h>
int soma3(){
	int resultado, num1, num2;
	printf("\n digite o primeiro numero: ");
	scanf("%d", &num1);
	printf("\n digite o segundo numero: ");
	scanf("%d", &num2);
	resultado = num1 + num2;
	return resultado;
}
int main(int argc, char *argv[]) {
	int res;
	res = soma3();
	printf("\n soma: %d", res);

	return 0;
}
```

Sub-rotinas com passagem de parâmetro

Passagem de parâmetro por valor

- Trabalha com **cópia** dos valores passados no momento da chamada

Passagem de parâmetro por valor

```
int soma(int num1, int num2){
        return num1 + num2;
}


int main(int argc, char *argv[]) {
        int res;
        res = soma(2,3);
        return 0;
}
```

Passagem de parâmetro por valor

- Modificação no parâmetro real não são refletidas no parâmetro formal

```
int soma_dobro(int num1, int num2) {
...
}
```

```
public static void main(String[] args) {
   int a = 1;
   int b = 2;
   Soma_dobro(a,b);
}
```

http://www.hinodelifestyle.com.br/wp-content/uploads/2015/01/HINODE-Perguntas-Frequentes-HINODE.png

Passagem de parâmetro por valor

- Qualquer alteração efetuada no parâmetro real, não afetará a *variável local*

```
public static void main(String[] args) {
   int a = 1;
   int b = 2;
   int resultado = soma_dobro(a,b);
}
```

Passagem de parâmetro por valor

```
int soma_dobro(int num1, int num2){

            num1 = 2 * num1;
            num2 = 2 * num2;
            return num1 + num2;
}
int main(int argc, char *argv[]) {
      int resultado, num1, num2;
      num1 = 2;
      num2 = 3;
      resultado = soma_dobro(num1,num2);

      printf("\n o resultado é: %d ", resultado);
      printf("\n o valor de num1  é: %d", num1);
      printf("\n o valor de num2 é: %d",num2);
      return 0;
}
```

Passagem de parâmetro por referência

- Parâmetros passados para uma função correspondem a endereços de memória ocupados por variáveis

Passagem de parâmetro por referência

- Qualquer alteração no parâmetro formal implica necessariamente no parâmetro real

```
int soma_dobro(int *num1, int *num2) {
...
}


public static void main(String[] args) {
   int a = 1;
   int b = 2;
   int resultado = soma_dobro(&a,&b);
}
```

Passagem de parâmetro por referência

```
int soma_dobro(int *num1, int *num2){

                *num1 = 2 * (*num1);
                *num2 = 2 * (*num2);
                 return *num1 + *num2;
}
int main(int argc, char *argv[]) {
       int resultado, num1, num2;
num1 = 2;
num2 = 3;
resultado = soma_dobro(&num1,&num2);

printf("\n o resultado é: %d ", resultado);
printf("\n o valor de num1  é: %d", num1);
printf("\n o valor de num2 é: %d",num2);

getchar();
       return 0;
}
```

Sub-rotina com passagem de parâmetro e sem retorno

- Recebem valores (parâmetros)
- Não retorna valor

Sub-rotina com passagem de parâmetro e sem retorno

```
void soma2(int num1,int num2){
	int resultado;
	resultado = num1 + num2;
	printf("\n a soma é: %d", resultado);
}

int main(int argc, char *argv[]) {
	soma (3,4);
	return 0;
}
```

Sub-rotina com passagem de parâmetro e com retorno

- Recebe valor no momento da chamada
- Retorna valor

Sub-rotina com passagem de parâmetro e com retorno

```
int soma(int num1, int num2){
	return num1 + num2;
}
int main(int argc, char *argv[])
{
	int res;
	res = soma4(2,3);
	return 0;
}
```

Sub-rotina em arquivo separado

```
#include <stdio.h>
#include "subrotina.h”

int main(int argc, char *argv[]) {
int resultado, num1, num2;
num1 = 2;
num2 = 3;
resultado = soma(num1,num2);
imprime();
printf("\n o resultado é: %d ", resultado);
printf("\n o valor de num1  é: %d", num1);
printf("\n o valor de num2 é: %d",num2);

getchar();
return 0;
}
```

sub-rotinas em arquivo separado

subrotina.h

```
void imprime(){
	printf("olá mundo!");
}

int soma(int numero1, int numero2){
	return numero1 + numero2;
}
```