SENAI CIMATEC

SISTEMA FIEB

TECNOLOGIA E INOVAÇÃO EM PROL DA INDÚSTRIA

SENAI CIMATEC

SISTEMA FIEB

Técnico em informática

SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

Programação de Aplicativo – 140h

Profª Francisleide Almeida

SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

Funções em C

Baseado em Profª Ms. Engª Elaine Cecília

Introdução

- Os programas C normalmente são escritos combinando-se novas funções com funções pré-definidas, disponíveis na biblioteca-padrão de C.
- A biblioteca padrão de C oferece uma rica coleção de funções para realização de cálculos matemáticos comuns, manipulação de strings, manipulação de caracteres, entrada/saída e muitas outras operações úteis.
- Isso torna seu trabalho mais fácil, pois essas funções oferecem muitas das capacidades de que você precisa.

Introdução

- Funções são chamadas, ou invocadas, por uma chamada de função, que especifica o nome da função e oferece informações, como argumentos, de que a função chamada precisa para realizar sua tarefa designada.
- Analogia: chefe e subordinado. O chefe é a função chamadora, o subordinado é a função chamada. O chefe pede a um subordinado que realize uma tarefa e informe quando ela tiver sido concluída. O chefe não sabe exatamente como o subordinado realiza suas tarefas. O subordinado pode chamar outros para ajudá-lo a realizar as tarefas sem o chefe saber.

Introdução

- As funções permitem a criação de um programa em módulos. Todas as variáveis descritas nas definições de função são **variáveis locais** – elas são conhecidas apenas na função em que são definidas.
- A maioria das funções possui uma lista de **parâmetros** que oferecem meios de transmissão de informações entre as funções. Os parâmetros de uma função **também são variáveis locais** dessa função.
- Nos programas que contém muitas funções, MAIN, normalmente é implementada como um grupo de chamadas para funções que realizam a maior parte do trabalho no programa.

Introdução

- Evite reinventar a roda. Quando possível, use as funções da biblioteca-padrão de C em vez de escrever novas funções. Isso pode reduzir o tempo de desenvolvimento do programa.
- O uso das funções das bibliotecas padrão de C ajuda a tornar os programas mais portáteis.
- Cada função deve ser limitada a realizar uma única tarefa bem sucedida, e o nome dela deve expressar essa tarefa. Isso facilita a abstração e promove a reutilização do software.
- Se você não puder escolher um nome curto que expresse o que a função faz, é possível que sua função esteja tentando realizar muitas tarefas diversas. Normalmente, é melhor quebrar essa função em várias funções menores – às vezes chamamos isso de decomposição.

Introdução

- Uma função é um conjunto de instruções desenhadas para cumprir uma tarefa particular e agrupadas numa unidade com um nome para referenciá-la.
- Funções dividem grandes tarefas de computação em tarefas menores, e permitem às pessoas trabalharem sobre o que outras já fizeram, em vez de partir do nada.
- Uma das principais razões para escrever funções é permitir que todos os outros programadores C a utilizem em seus programas.
- Funções apropriadas podem frequentemente esconder detalhes de operação de partes do programa que não necessitam conhecê-las.

Introdução

- • Você já usou a função printf() sem conhecer detalhes de sua programação.
- A existência de funções evita que o programador tenha de escrever o mesmo código **repetidas vezes**.
- Qualquer sequencia de instruções que apareça mais de uma vez no programa é candidata a ser uma função.
- O código de uma função é agregado ao programa uma única vez e pode ser executado muitas vezes no decorrer do programa.
- O uso de funções **reduz o tamanho do programa**.

Funções da biblioteca matemática

- Ex:

função	descrição	exemplo
sqrt(x)	raiz quadrada de x	sqrt(900,0) = 30,0
exp(x)	exponencial e^x	exp(1,0) = 2,718
log(x)	logarítimo natural de x (base e)	log(2,718) = 1,0
log10(x)	logarítimo de x (base 10)	lo10(1,0) = 0,0
fabs(x)	valor absoluto de x	fabs(-13,5) = 13,5
ceil(x)	arredonda x ao menor inteiro não menor que x	ceil(9,2) = 10,00 ceil(-9,8) = 9,00
floor(x)	arredonda x ao maior inteiro não maior que x	floor(9,2) = 9,00 floor(-9,8) = 10,00
pow(x,y)	x elevado à potência y (x^y)	pow(2,7) = 128,0
fmod(x,y)	módulo, resto, de x/y como um número em ponto flutuante	fmod(13,657, 2,33) = 1,992
sin(x)	Seno trigonométrico de x (x em radianos)	Sin(0,0) = 0,0

Definição

- Main: função principal que chama funções da biblioteca padrão para realizar suas tarefas.
- Exemplo de programa que utiliza função: A função square é chamada ou invocada em main dentro da instrução printf.

A função square recebe uma cópia do valor de x no parâmetro y. Depois, square calcula y*y. O resultado é passado de volta à função printf em main, onde square foi chamada, e printf exibe o resultado. O processo é repetido 10 vezes, usando for.

```
/* Criando e usando uma função definida pelo programador */

//cabeçalhos
#include <stdio.h>

//protótipo da função
int square(int y);

//função principal
main()
{
    int x;//contador
    //loop: calcule e exiba o quadrado de x 10 vezes
    for(x=1; x<=10; x++){
       printf("%d ", square(x));//chamada da função
       printf("\n");
    }//fim do for
    system("PAUSE");
}//fim da função principal

/*definindo a função de quadrado de x
y é um parâmetro, uma cópia do argumento à função
ou seja y = x*/
int square(int y){
    return y*y;//retorna o quadrado de y como um inteiro;
}//fim da função
```

- A definição da função square mostra que square espera por um parâmetro inteiro y. A palavra-chave int antes do nome da função indica que square retorna um resultado inteiro. A instrução return dentro de squar passa o resultado do cálculo de volta à função chamadora;
- FORMATO DE UMA DEFINIÇÃO DE FUNÇÃO:

```
Tipo-valor-retorno nome-função( lista-de-parametros){
        definições instruções
}
```

Dicas

- Esquecer de retornar um valor de uma função que deveria retornar um valor pode gerar erros inesperados.
- Retornar um valor de uma função com um tipo de retorno void é um erro de compilação.
- Especificar parâmetros de função do mesmo tipo como double x, y e vez de double x, double y, resulta em um erro de compilação.
- Colocar um ponto e vírgula após o parêntese à direita que delimita a lista de parâmetros de uma definição de função é um erro de sintaxe.
- Definir, novamente, um parâmetro como uma variável local em uma função é um erro de compilação.

Dicas

- Embora não seja errado, não use os mesmos nomes para os argumentos de uma função e para os parâmetros correspondentes na definição da função. Isso ajuda a evitar ambiguidades.
- As definições e instruções dentro das chaves formam o corpo da função. O corpo da função também é chamado de bloco. As variáveis podem ser declaradas em qualquer bloco, e os blocos podem ser aninhados. Uma função não pode ser definida dentro de outra função.
- Escolher nomes de função e de parâmetro significativos torna os programas mais legíveis e evita o uso excessivo de comentários.

Dicas

- Geralmente, uma função não deve ocupar mais que uma página. Melhor ainda, as funções não devem ocupar mais que meia página. Funções pequenas promovem a reutilização de software.
- Os programas devem ser escritos como coleções de pequenas funções. Isso os torna mais fáceis de serem escritos, depurados, mantidos e modificados.
- Uma função que exige um grande número de parâmetros pode estar realizando tarefas demais. Considere dividi-la em funções menores, que realizem as tarefas separadamente. O cabeçalho da função deverá caber em uma linha, se possível.

SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

Exemplos

```c
//Esse programa encontra o máximo de três inteiros
#include <stdio.h>
int maximum(int x, int y, int z);
int main(void)
{
      //variáveis que armazenarão os números
      int number1;
      int number2;
      int number3;
      //solicitando os números
      printf(" \n Digite tres numeros inteiros \n");
      //lendo os números digitados
      scanf("%d%d%d", &number1, &number2, &number3);
      //number1, number1, number3 são argumentos da chamada da função maximum
      printf("Maximo e: %d \n ", maximum(number1, number2, number3), " \n ");
      system("PAUSE");
      return 0;
}//fim da função principal
//definição da função maximum: x, y e z são parâmetros
int maximum(int x, int y, int z ){
    int max=x;//considera que x é o maior
    if( y>max ){//se y é maior que max, atribui y a maior
       max = y;
    }
    if( z>max ){ //se z é maior que max, atribui z a maior
        max = z;
    }
    return max; // max é o maior valor
}//fim da função
```

Exemplos

```c
//Mostra a escrita da função celsius()
#include <stdio.h>
#include <stdlib.h>
float celsius(float); //protótipo da função
int main() //função principal
{
    float c, f;
    printf("Digite a temperatura em graus Fahrenheit: \n");
    scanf("%f", &f);
    c=celsius(f); //chamada á função
    printf("Celsius = %.2f \n", c);
    system("PAUSE");
    return 0;
}//fim da função principal
float celsius(float fahr)
{//função celsius
      float c;
      c=((fahr-32.0)*5.0)/9.0;
      return c;
}//fim da função
```

Protótipos de funções

- O protótipo de função diz ao compilar o tipo de dado retornado pela função, o número de parâmetros que a função espera receber, os tipos dos parâmetros e a ordem em que esses parâmetros são esperados.
- O compilador utiliza protótipos de função para validar as chamadas de função.
- As versões anteriores de C não realizavam esse tipo de verificação, de modo que era possível chamar funções incorretamente sem que o compilador detectasse os erros.
- Essas chamadas poderiam resultar em erros fatais no tempo de execução, ou em erros não fatais que causavam erros lógicos sutis e difíceis de detectar.
- Os protótipos de funções corrigem essa deficiência.

Protótipos de funções

- Inclua protótipos de função em todas as funções para tirar proveito das capacidades de verificação de tipo da linguagem C.
- Utilize diretivas do pré-processador #include para obter protótipos de função para as funções da biblioteca padrão a partir dos cabeçalhos para as bibliotecas apropriadas, ou para obter cabeçalhos que contenham protótipos de função para funções desenvolvidas por você e/ou pelos membros do seu grupo.
- Às vezes, os nomes de parâmetros são incluídos nos protótipos de função, nossa preferência, para fins de documentação. O compilador ignora esses nomes.
- Esquecer de colocar o ponto e vírgula ao final de um protótipo de função é um erro de sintaxe.
- Uma chamada de função que não corresponde ao protótipo de função consiste em um erro de compilação.

Protótipos de funções

- Uma função não pode ser chamada sem antes ter sido declarada.
- A declaração de uma função é dita protótipo da função, é uma instrução geralmente colocada no início do programa que estabelece o tipo da função e os argumentos que ela recebe.
- O protótipo da função permite que o compilador verifique a sintaxe de chamada à função.
- O propósito principal de escrita de protótipos de funções em C é o de fornecer ao compilador as informações necessárias sobre o tipo da função, o número e o tipo dos argumentos.
- Assim, tornamos possível a verificação da sintaxe de chamada à função.
- Sem o protótipo da função, o compilador não tem como verificar e checar se há erros em seu uso.

Conversão de tipo

- Os valores de argumentos que não correspondem exatamente aos tipos de parâmetro no protótipo de função são transformados no tipo apropriado antes que a função seja chamada.
- Essas conversões podem gerar resultados incorretos se as regras de promoção não forem seguidas. Essas regras especificam como os tipos podem ser convertidos para outros sem que haja perda de dados.
- Um int é convertido automaticamente para double sem mudar seu valor. Porém, um double convertido para int trunca a parte fracionária do valor double.
- Converter tipos inteiros grandes para tipos inteiros pequenos também pode resultar em valores alterados

Conversão de tipo

- As regras de promoção se aplicam automaticamente a expressões que contenha valores de dois ou mais tipos de dados, ou expressões de tipo misto.
- O tipo de cada valor em uma expressão de tipo misto é automaticamente promovido para o tipo mais alto na expressão – na realidade, uma versão temporária de cada valor é criada e usada na expressão, os valores originais permanecem inalterados.
- A conversão de valores em tipos inferiores normalmente resulta em um valor incorreto. Portanto, um valor pode ser convertido em um tipo inferior somente pela atribuição explícita do valor a uma variável do tipo inferior, ou usando-se um operador de coerção.

Conversão de tipo

- Os valores de argumentos de função são convertidos para tipos de parâmetro de um protótipo de função como se estivessem sendo atribuídos diretamente às variáveis desses tipos.
- Converter um tipo de dados mais alto na hierarquia de promoção em um tipo inferior pode alterar o valor do dado. Muitos compiladores emitem advertência nesses casos.
- Sempre inclua protótipos de função nas funções que você define ou usa em seu programa; isso ajuda a evitar erros e advertências na compilação
- Um protótipo de função colocado fora de qualquer definição de função se aplica a todas as chamadas para a função que aparecem após o protótipo de função no arquivo.

Conversão de tipo

- Um protótipo de função colocado dentro de uma função se aplica apenas às chamadas feitas nessa função.
- Se não há protótipo de função par a uma função, o compilador forma seu próprio protótipo, usando a primeira ocorrência da função – ou a definição de função, ou uma chamada para a função. Normalmente, isso causa advertência ou erros, a depender do compilador.

Chamada por valor e referência

- Existem duas formas de se chamar funções, a chamada por valor e a chamada por referência.
- Quando os argumentos são passados por valor, uma cópia do valor do argumento é feita e passada para a função chamada.• As mudanças na cópia não afetam o valor original da variável na chamadora.
- Quando um argumento é passado por referência, o chamador permite que a função chamada modifique o valor da variável original.

Chamada por valor e referência

- A chamada por valor deverá ser usada sempre que a função chamada não precisar modificar o valor da variável original da chamadora.
- Isso evita efeitos colaterais acidentais que tanto atrapalham o desenvolvimento de sistemas de software corretos e confiáveis.
- A chamada por referência deve ser usada apenas nos casos de funções chamadas confiáveis, que precisam modificar a variável original. Em C, todas as chamadas são feitas por valor.

O tipo de uma função

- O tipo de uma função é definido pelo tipo de valor que ela retorna por meio do comando return.
- Uma função é do tipo float quando retorna um valor do tipo float.
- Os tipos de funções C são os mesmos tipos que o das variáveis, exceto quando a função não retorna nada.
- Nesse caso, ela é do tipo void.
- O tipo de uma função é determinado pelo valor que ela retorna via comando return, e não pelo tipo de argumentos que ela recebe.

O comando return

- O comando return termina a execução da função e retorna o controle para a instrução seguinte do código de chamada.
- Se, após a palavra return, houver uma expressão, o valor desta é retornado à função que chama.
- Esse valor é convertido para o tipo da função, especificado no seu protótipo.
- A sintaxe de uma instrução return tem uma das três seguintes formas: return; return expressão; return (expressão);

O comando return

- Funções do tipo void podem ter um comando return desacompanhado de expressão.
- Nesse caso, o comando return serve para terminar a execução da função.
- Em função do tipo void, o comando return não é obrigatório.
- Uma função sem comando rturn termina quando encontra a chave de fechamento ({)
- Enquanto vários valores podem ser passados para uma função como argumentos, não é permitido o retorno de mais de um valor por meio do comando return.
- O comando return pode retornar somente um único valor para a função que chama.

```c
#include <stdio.h>
#include <stdlib.h>
const float PI=3.14159;
float area(float);
main()
{
      float raio;
      printf("Digite o raio da esfera \n ");
      scanf("%f", &raio);
      printf("A area da esfera e %.2f \n ", area(raio));
      system("PAUSE");
      return 0;
}
float area(float r)
{
      return(4*PI*r*r);
}
```

Atividades