Laboratório de Programação - Linguagem C

Prof. Cezar Macegoza

Diretiva #define

A diretiva #define serve para definir constantes simbólicas que aumentam a legibilidade do código-fonte. Essa diretiva associa um identificador a um texto da seguinte maneira:

#define identificador texto

Diretiva #define

```
#include <stdio.h>

#define diga  printf

#define oi “\nOlá, tudo bem?”

   int main() {

      diga(oi);

}
```

Diretiva #define

```
#include <stdio.h>

#define quad(n) n *n

#define diga printf


int main()

{

    int resultado;

    resultado = quad(4);

        diga("%d",resultado);

}
```

Diretiva #include

Essa diretiva, quando executada, faz com que uma cópia do arquivo cujo nome é dado entre < e > seja incluído no código-fonte. Por exemplo, suponha que definimos as macros a seguir e as salvamos num arquivo denominado **macros.h**

```
#define quad(n) ( (n)*(n) )

#define abs(n) ( (n)<0? –(n) : (n) )

#define max(x,y) ( (x)>(y) ? (x) : (y) )
```

Diretiva #include

```
#include <stdio.h>

#include "macros.h"

int main()

{

    int a, b;

    printf("\nDigite dois números: ");

    scanf("%d",&a );

    scanf("%d", &b);

    printf("\nO máximo é %d!", max(a, b));

}
```

Diretiva #include

Duas observações devem ser feitas a respeito da inclusão de arquivos:

- Qualquer arquivo, com qualquer extensão, pode ser incluído num programa fonte através da diretiva #include.

- A notação < e > é preferencialmente utilizada para arquivos de inclusão padrão da linguagem C. Para incluir arquivos definidos pelo usuário, utilize a notação '' e '', conforme se observa no programa acima.

Exercício

1. Crie um arquivo com as macros da fórmula de báskara(equação do segundo grau)e faça um programa que use esse arquivo para testar essas macros.

Funções

Para definir uma função, empregamos a seguinte forma básica:

```
tipo nome(parâmetros) {
   declarações
   comandos
}
```

sendo que:

- **tipo** refere-se ao tipo de resposta que a função devolve e deve ser void (vazio) se a função não tem valor de resposta;

- **nome** é o identificador da função no resto do programa;

- **parâmetros** é uma lista de variáveis que representam valores de entrada para a função e deve ser void caso não haja valores de entrada;

- dentro do corpo da função, a primeira seção é destinada à **declaração** das variáveis e a segunda, aos **comandos**.

Funções que não devolvem resposta

```
#include <stdio.h>

void Mensagem (void);

int main ()

{

    Mensagem();

    printf ("Diga de novo:\n");

    Mensagem();


}

void Mensagem (void)

{

    printf ("Ola! Eu estou vivo.\n");

}
```

Funções que devolvem resposta

Se uma função não é do tipo void, então ela deve, necessariamente, devolver um valor como resultado de sua execução. Para isso, a função deve empregar o comando return. Esse comando, além de especificar a resposta da função, faz com que o controle retorne ao ponto onde ela foi chamada no programa, interrompendo imediatamente sua execução.

Funções que devolvem resposta

```
#include <stdio.h>
#include <math.h>
float hip (float a, float b);
int main ()
{
    float a, b,resultado;
    printf("Escreva um lado do triângulo retângulo\n");
    scanf("%f",&a);
    printf("Escreva o outro lado do triângulo retângulo\n");
    scanf("%f",&b);
    resultado = hip(a,b);
    printf("O resultado é: %f \n",resultado);

}
float hip(float a, float b) {
   float h;
   h = sqrt(pow(a,2)+pow(b,2));
   return h;
}
```

Exercício

2. Codifique a função fat(n), que devolve o fatorial de n.

Exercício

3. Codifique a função pot(x,n), que devolve x elevado a n.

Recursividade

A recursividade é um princípio que nos permite obter a solução de um problema a partir da solução de uma instância menor de si mesmo. Para aplicar esse princípio devemos supor que a solução da instância menor é conhecida.

Para ser útil, uma função recursiva deve ter um ponto de parada, ou seja, deve ser capaz de interromper as chamadas recursivas e executar em tempo finito.

Recursividade

Ao definir uma função recursiva devemos identificar:

1. A base da recursão, i.e. a instância mais simples do problema em questão.

2. O passo da recursão, i.e. como simplificar o problema em questão.

Exemplo

```
#include <stdio.h>
int potencia(int base, int expoente)
{
    if (expoente == 0)
        return 1;
    else
        return (base * potencia(base, expoente - 1));
}
int main()
{
    int base, expoente, resultado;
    printf("Escreva a base do número\n");
    scanf("%d", &base);
    printf("Escreva a expoente do número\n");
    scanf("%d", &expoente);
    resultado = potencia(base, expoente);
    printf("%d elevado a %d = %d\n", base, expoente, resultado);
    return 0;
}
```

Exercício

4. Seja a somatória abaixo. Faça uma função recursiva para realizar o cálculo.

$$\sum_{i=1}^{n} = i^2$$

Exercício

6. Calcular o fatorial de um número natural.