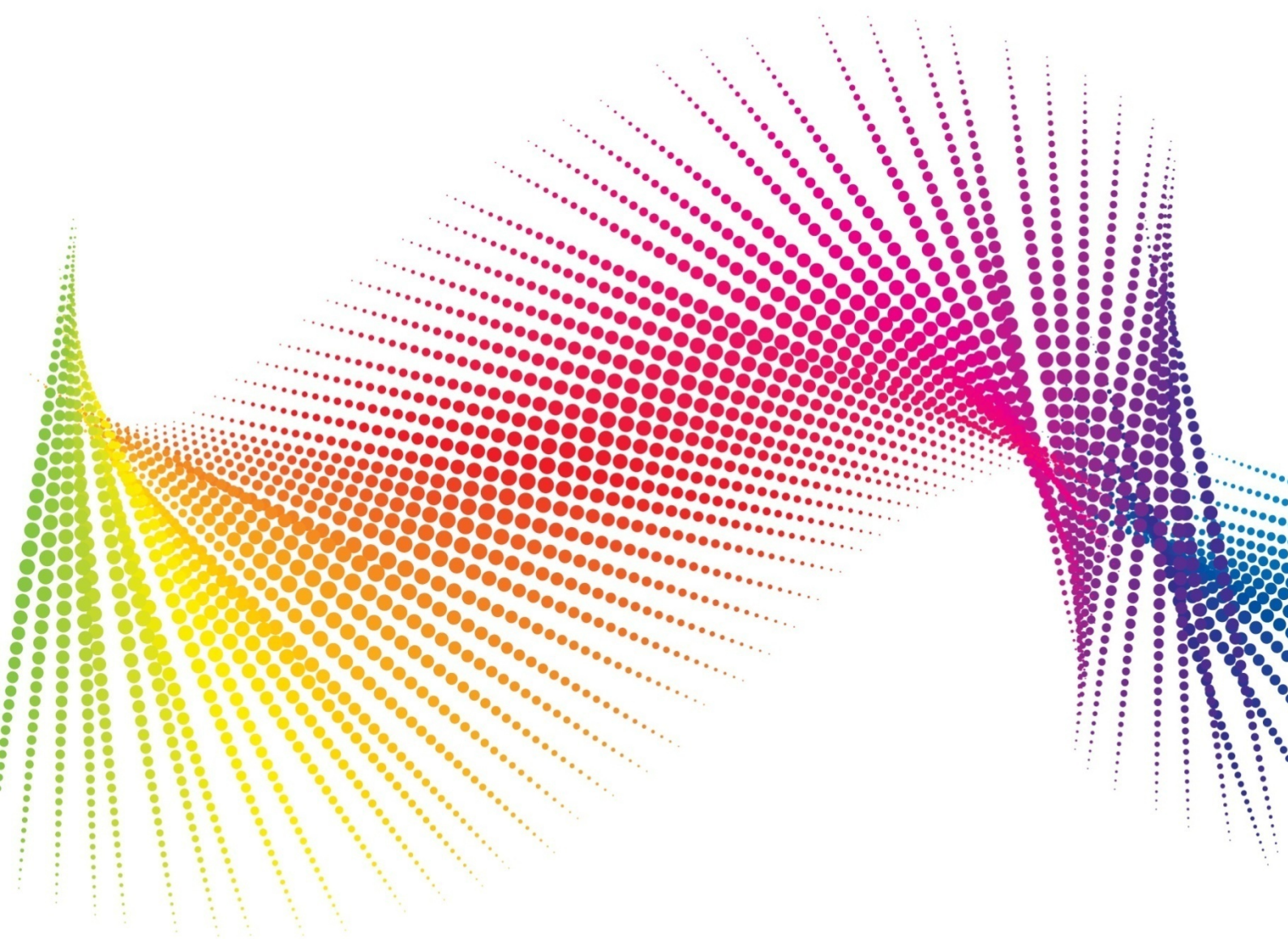


Estrutura de Dados

Aula 08



Este material é parte integrante da disciplina oferecida pela UNINOVE.

O acesso às atividades, conteúdos multimídia e interativo, encontros virtuais, fóruns de discussão e a comunicação com o professor devem ser feitos diretamente no ambiente virtual de aprendizagem UNINOVE.

Uso consciente do papel.

Cause boa impressão, imprima menos.

Aula 8: Funções

Objetivo: Continuar a estudar as funções na linguagem C, explorando o escopo de variáveis.

Regras de escopo de variáveis

Escopo de uma variável (ou constante) é o termo técnico que define a região do programa fonte C em que a variável ou constante é reconhecida ou está ativa.

A duração de uma variável se refere ao tempo de vida de armazenamento desta na memória principal. Pelas regras de escopo, uma variável (ou constante) pode ser local ou global.

Variáveis locais ou internas

As variáveis locais ou internas são declaradas e válidas dentro de um bloco de código.

Na linguagem C um bloco de código é uma porção de código delimitada por um abre-chave “{” e um fecha-chave “}”. Assim, elas só podem ser referenciadas dentro do bloco de código em que tenham sido declaradas, não sendo reconhecidas fora dele.

Estas variáveis devem ser declaradas no início do bloco de código, antes de qualquer comando. Elas existem apenas enquanto o seu bloco de código está sendo executado: são criadas na entrada e destruídas na saída, não retendo, portanto, seus valores entre chamadas. Desta forma, podem receber um endereço de memória diferente a cada vez que o bloco é executado.

Exemplos:

Aqui variável x é válida somente dentro da função func1().

```
float func1(void)
```

```

{ float x;
x = 10;
return (sqrt(x));
}

```

Aqui variável x é válida somente dentro da função func2() e não é a mesma da função func1(), embora tenha o mesmo nome e seja do mesmo tipo.

```

float func2(void)
{ float x;
x = 200.10;
return(x);
}

```

Em outras palavras: x em func1 não tem absolutamente qualquer relação com x de func2, mas ambos são variáveis locais

Aqui, a variável j é local ao bloco de comandos for apenas.

```

for (i=0; i<10; i++)
{int j = 2;
printf("%d\n",j*i);
}

```

Assim, j só é visível dentro do corpo da instrução for. **Os parâmetros de entrada de uma função se comportam como variáveis locais.**

Exemplo de um programa para ilustrar o uso de variáveis locais

/*programa pot.c Calcula as potências de um número real */

```
#include <stdio.h>
```

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

```
float pot(float,int);
```

```
main()
```

```
{
```

```
float base,result;
```

Declarando um protótipo da função. Esta função recebe dois parâmetros; a base (float) e o expoente (int). Calcula o resultado e retorna o valor.

Criando variáveis locais (internas). Estas variáveis só existem aqui dentro desta função principal, por isto que elas têm que ser passadas para a função pot que vai fazer o cálculo.

Laço infinito. Quando você não quiser mais testar o programa é só digitar zero que sai do laço pelo comando break.

```
int exp;
while(1){
    printf("\n Entre com a base (entre com zero para desistir:");
    scanf("%f",&base);
    if(base==0)
        break;
    printf("\n Entre com o expoente:");
    scanf("%d",&exp);
    result = pot(base,exp);
    printf("\n %.4f elevado a %d = %.4f\n",base,exp,result);
    system("pause");
}
}
```

O comando scanf pega a base que foi digitada. Se o valor digitado for zero então o comando break faz sair do laço e para a entrada de dados então.

Após pegar a base e o expoente, é chamado a função pot e passa-se a base e o expoente como parâmetros. A função pot calcula o resultado e retorna o valor para a variável result. Logo, é impresso o resultado.

O código da função pot está aqui. Ela recebe a base e o expoente. Observe o tipo de retorno da função (float). Ela retorna o resultado que é um valor real.

```
float pot(float base, int exp)
```

```
{
    float result=1;
    int i;
    if(exp > 0)
        for(i=0;i<exp;i++)
            result=result * base;
    else for(i=exp;i<0;i++)
        result=result * 1/base;
    return result;
}
```

Se o expoente for positivo, ele entra aqui.

Aqui fazemos o cálculo da potenciação através de um laço, o qual multiplica a base por ela mesma o número de vezes o expoente. Observe a lógica como é. Faça um teste de mesa disto e procure entender.

O laço para o cálculo do expoente negativo é assim. Observe que agora somamos + 1 a um número negativo até ele chegar em zero, e o cálculo é feito pelo inverso da base.

Se o expoente for negativo entra aqui

O resultado expresso aqui na variável result tem que ser retornado porque esta variável só existe aqui. Mesmo que tenhamos declarado outra variável na função principal com o mesmo nome, não importa. Esta variável só existe aqui dentro da função pot.

Caro aluno, agora acesse o AVA e teste seus conhecimentos entreterendo-se com o jogo de forca sobre funções.

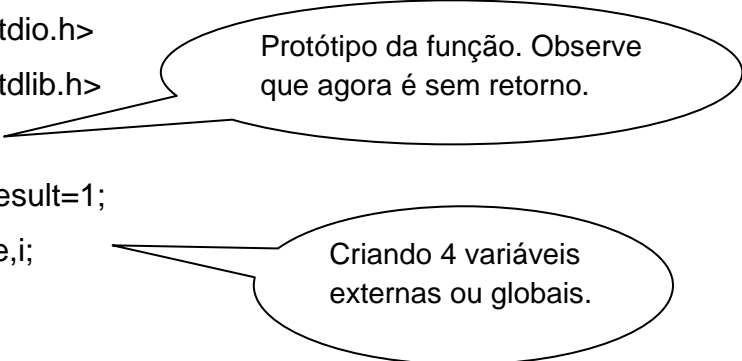
Variáveis Globais

As variáveis globais são declaradas fora de qualquer função, em qualquer lugar, anterior à primeira vez em que são usadas e elas valem em qualquer lugar abaixo de onde foram criadas. Assim, podem ser usadas em qualquer bloco de código.

Exemplo do mesmo programa para ilustrar agora o uso de variáveis globais

```
#include <stdio.h>
#include <stdlib.h>
void pot();
float base,result=1;
int expoente,i;
main()
{
    while(1){
        printf("\n Entre com a base (entre com zero para desistir:");
        scanf("%f",&base);
        if(base==0)
            break;
        printf("\n Entre com o expoente:");
        scanf("%d",&expoente);
        pot();//chama a função sem passar e receber nada
        printf("\n %.4f elevado a %d = %.4f\n",base,expoente,result);
        system("pause");
    }//fim while
}

void pot()
```



```
{  
    result=1;  
    if(expoente > 0)  
        for(i=0;i<expoente;i++)  
            result=result * base;  
    else for(i=expoente;i<0;i++)  
        result=result * 1/base;  
}
```

Obs: veja que agora a função foi declarada com o tipo sem retorno (void) porque não há necessidade alguma de passar os parâmetros por funções e nem de retornar valores, visto que as variáveis foram todas declaradas globais ou externas. Os resultados podem ser impressos em qualquer lugar porque as variáveis globais valem em qualquer lugar da função diferentemente do programa exemplo anterior.

Embora o seu uso seja mais fácil, existem os seguintes pontos a serem considerados: as variáveis globais só devem ser usadas quando se necessita de um mesmo dado em muitas funções do programa, pois o seu uso tem pontos negativos já que ocupam memória durante todo o tempo de execução do programa e podem gerar a ocorrência de erros devido a efeitos colaterais além de tornar mais difícil a depuração do programa.

Importante: se uma variável local é declarada em um bloco de código com o mesmo nome que uma variável global, qualquer referência a este nome dentro do bloco de código em que foi declarada diz respeito à variável local e não à global.

Agora, caro aluno, vamos praticar resolvendo os exercícios propostos. **Leia a lista, resolva os exercícios e verifique seu conhecimento.** Caso fique alguma dúvida, leve a questão ao Fórum e divida com seus colegas e professor.

REFERÊNCIAS

MIZRAHI, V.V. *Treinamento em linguagem C*. Módulo 1. 2 ed. São Paulo: Makron Books, 2006.

_____. *Treinamento em linguagem C. Módulo 2.* 2 ed. São Paulo: Makron Books, 2006.

SCHILDT, H. *C Completo e total.* 3. ed. São Paulo: Makron Books, 1997.