Estrutura de Dados

Aula 1 : Revisão de Ponteiros

Prof. MSc. Fausto Sampaio

fausto.cefet@gmail.com

Centro Universitário UniFanor - Wyden

5 de novembro de 2019

Sumário

Linguagem C

Linguagem C

- C é uma linguagem de médio nível; Compilada;
- Médio nível != ser menos poderosa, difícil de usar ou menos desenvolvida;
- C = elementos(Alto Nível) + elementos(Baixo Nível).

Nível	Linguagem
Alto	Ada
	Modula-2
	Pascal
	COBOL
	FORTRAN
	BASIC
Médio	C
	C++
	FORTH
Baixo	Macro-Assembler
	Assembler

Forma de um Programa em C

```
Declarações_globais;
tipo_devolvido main(lista_de_parâmetros){
    sequência_de_comandos;
}
tipo_devolvido f1(){
    sequência_de_comandos;
}
tipo_devolvido f2(){
    sequência_de_comandos;
}
.
.
tipo_devolvido fN(){
    sequência_de_comandos;
}
```

Exemplo de Código em C

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

#define PI 3.14

float calc_area(float r);

int main(){
    float raio;
    printf("Digite o raio da circunferencia:\n");
    scanf("%f",&raio);
    printf("A area eh: %.2f", calc_area(raio));
}


float calc_area(float r){
    return PI * r * r;
}
```

Ponteiros

Ponteiros

Razão de uso

- Permitem mudar os argumentos das funções;
- Manipular as rotinas de alocação dinâmica;
- Aumentar a eficência do programa;
- Criação de estruturas de dados como listas encadeadas e árvores.

Definição

- Ponteiro é uma variável que contém um endereço de memória.
- O conteúdo dessa variável é a posição de outra variável na memória.
- Assim, um ponteiro aponta para outra variável quando contém o endereço desta.

Declaração I

Declarar

$< tipo > *nome\_identificador;$

Inicializar

$nome\_identificador = NULL;$

Operador: &

O & é um operador que devolve o endereço da memória do seu operando.
Exemplo: [m = &count;]
Esse endereço é a posiçao interna da variável na memória do computador.
Não tem relacão nenhuma com o valor de count.

Declaração II

Operador: *

O $*$ é um operador que devolve o valor da variável localizado no endereço que o segue.

q = *m;

Operador: *

```
int main()
{
int numero = 5; /* suponha na posicao 1000 */
int *p = &numero; /* p aponta pra numero */
printf("numero = %d", *p);
}
```

Alocação Dinâmica I

Definição

É o meio pelo qual se obtém memória em tempo de execucão.

Alocação Dinâmica II

Funções de Alocação Dinâmica

As funções de alocação dinâmica mais comuns são: **malloc()** e **free()**.

Função: malloc()

void *malloc(size_t numero_bytes);

Exemplo 1: malloc()

```
char *p;
p = malloc(1000); /*obtém mil bytes*/
```

Exemplo 1: malloc()

```
int *p;
p = malloc(50 * sizeof(int)); /*espaço p/ 50 inteiros*/
```

Alocação Dinâmica III

A memória é infinita?

```
int *p;
p = malloc(50 * sizeof(int));
if(!p){
     printf("Sem memoria!!!");
     exit(1);
}
```

Função free()

```
/*libera memoria, oposto de malloc()*/
void free(void *p);
```

Revisão de Lógica de Programação

Algoritmos

Definição

Um **algoritmo** é uma sequência lógica de instruções que devem ser seguidas para a resolução de um problrma ou tarefa. (Sandra Puga, 2004)

Dia-a-dia

- Preparar um bolo;
- Trocar o pneu do carro;
- Trocar uma lâmpada;

Problemas computacionais

- Construção de Interfaces;
- Softwares e Hardware;
- Planejamento de redes;
- Documetação de Sistemas;

Sequencial

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

int main(){
    float n1, n2;
    printf("Digite duas notas:\n");
    scanf("%f",&n1);
    scanf("%f",&n2);
    printf("A media eh: %.2f", (n1+n2)/2);

    return 0;
}
```

Controle: Seleção, Decição ou Desvio

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

int main(){
    int idade;
    printf("Digite sua idade:\n");
    scanf("%d",&idade);
    if(idade >= 18){
        printf("Aluno maior de 18 anos.");
    }
    else{
        printf("Aluno menor de 18 anos.");
    }
    return 0;
}
```

Controle: Repetição com teste no início

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

int main(){
    int num;

    printf("Digite um numero inteiro:\n");
    scanf("%d",&num);

    while(num > 0){
        printf("%d\n",num);
        num = num - 1;
    }

    return 0;
}
```

Controle: Repetição com teste no fim

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

int main(){
    int num;

    do{
        printf("Digite um numero inteiro positivo:\n");
        scanf("%d",&num);
    }while(num < 0);

    return 0;
}
```

Controle: Repetição com variável de controle

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

int main(){
    int num;

    printf("Digite um numero inteiro:\n");
    scanf("%d",&num);

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

    return 0;
}
```

Operadores

Aritméticos

Símbolo	Significados
-	Subtração
+	Adição
*	Multiplicação
/	Divisão
%	Resto da divisão (módulo)

Relacionais e Lógicos

Operadores Relacionais

Operador	Ação
>	Maior que
>=	Maior que ou igual
<	Menor que
<=	Menor que ou igual
==	Igual
!=	Diferente

Operadores lógicos

Operador	Ação
&&	AND
||	OR
!	NOT