Aula 1 – Revisão de Funções e Ponteiros

Uso de Funções

Definição: conjunto de comandos agrupados em um bloco que recebe um nome e através deste pode ser ativado.

O uso de funções tem como características:

- ✓ Permitir o reaproveitamento de código já construído ou sua alteração de forma simples e rápida;
- ✓ Evitar que um trecho de código seja repetido várias vezes dentro de um mesmo programa;
- ✓ Evitar que os blocos do programa fiquem grandes demais, ocasionando a dificuldade de entendimento – facilita a leitura do programa-fonte.

Formatação Geral de uma Função em Linguagem C

```
tipo_da_funcao Nome_da_Funcao (Lista de Parâmetros){
 // corpo da função
 }
```

A Lista de Parâmetros também é chamada de Lista de Argumentos, é opcional (porém é obrigatório o uso dos parênteses, quando vazia).

Parâmetros

Os parâmetros possibilitam que se defina sobre quais dados à função deve operar. A Passagem de parâmetros pode ser:

- Por Valor: realiza uma cópia da variável;
- Por Referência: passa somente o endereço da variável (uso de ponteiros).

Para definir os parâmetros de uma função o programador deve explicitá-lo como se estivesse declarando uma variável, entre os parênteses do cabeçalho da função. Caso precise declarar mais de um parâmetro, basta separa-los por vírgulas.

No exemplo abaixo, temos a função soma que possui dois parâmetros, sendo o primeiro um float e o segundo um int.

```
float SOMA(float a, int b){
   float result;
   result = a + b;
   return(result);
   }
```

ou

```
float SOMA(float a, int b){
  return(a+b);
  }
```

Os parâmetros são passados para uma função de acordo com a sua posição, ou seja, o primeiro parâmetro atual (da chamada) define o valor do primeiro parâmetro formal (na definição da função), o segundo parâmetro atual define o valor do segundo parâmetro formal e assim por diante. Os nomes dos parâmetros na chamada não têm relação com os nomes dos parâmetros na definição da função.

Exemplo:

```
#include <stdio.h>

int produto(int k,int z){
 return(k*z);
 }

int soma(int x,int y){
 printf("Soma = %i\n",x+y);
 }

int main(){
  int x=10,y=5,r;
  r=produto(x,y);
  printf("Produto = %i\n",r);
  soma(x,y);
  }
```

Uso de Ponteiros

Da mesma forma que declaramos variáveis para armazenar valores, podemos declarar variáveis que, servem para armazenar valores de endereços de memória, os ponteiros.

Para declararmos um ponteiro, usamos o mesmo tipo com os nomes das variáveis precedidas pelo caractere *. Assim, podemos escrever: *int *p* para declarar um ponteiro p do tipo int (inteiro).

Usamos o operador unário & (“endereço de”), que resulta no endereço da posição de memória reservada para a variável. O operador unário * (“conteúdo de”), aplicado as variáveis do tipo ponteiro, acessa o conteúdo do endereço de memória armazenado pela variável ponteiro.

Exemplo de uso de ponteiros com um vetor: Calculo da média dos 10 elementos armazenados em um Vetor.

```
#include<stdio.h>
#include<stdlib.h>
#define TAM 10

int GeraVetor(int *Vet){
 int i;

 for(i=0;i<TAM;i++){
    Vet[i]=rand()%100;
    }
 }

float Media(int n, int *Vet){
  int i,s=0;
  for(i=0;i<n;i++){s+=Vet[i];}
  return (s/(float)n);
  }
```

```
int main(){
 int Vetor[TAM],i;
 GeraVetor(Vetor);
 printf("\nMedia do vetor eh = %.2f\n",Media(TAM,Vetor));
 }
```