Funções

- O tipo void

Em inglês, **void** quer dizer vazio e é isto mesmo que o **void** é.

Ele nos permite fazer funções que não retornam nada:

void nome_da_função (declaração_de_parâmetros);

Numa função, como a acima, não temos valor de retorno na declaração **return**. Aliás, neste caso, o comando **return** não é necessário na função. Contudo, podemos utilizá-lo em pontos onde desejamos que a função finalize sua execução.

Funções

Uma observação importante é que sempre que tivermos uma função com um tipo de retorno diferente de void, torna-se necessário retornar um valor com o comando return compatível com o tipo de retorno. Antes de tratarmos a main como uma função não utilizávamos o comando return em seu corpo, isso representa um comportamento inadequado, nos utilizamos dele devido ao não conhecimento da forma correta de se proceder.

Conforme podemos observar nas funções main() dos códigos fonte dos programas desenvolvidos até o momento, uma função pode não ter parâmetros.

UNIVASF

Extrapolando esta observação, podemos fazer funções como a presente no programa a seguir:

```
#include <stdio.h>
void Mensagem (void)
{
  printf ("Ola! Eu estou vivo.\n");
}
int main ()
{
  Mensagem();
  printf ("\tDiga de novo:\n");
  Mensagem();
  return 0;
}
```

Funções

- Escopo de variáveis

O escopo de variáveis é o conjunto de regras que determinam o uso e a validade de variáveis nas diversas partes do programa.

Veremos agora três tipos de variáveis, no que se refere ao escopo:

Funções

- Variáveis locais

Variáveis locais são aquelas que só têm validade dentro do bloco no qual são declaradas. **Podemos declarar variáveis dentro de qualquer bloco.**

Só para lembrar: um bloco começa quando abrimos uma chave e termina quando fechamos a chave. A declaração de variáveis locais é a primeira coisa que devemos colocar num bloco.

UNIVASF

```
func1 (...)
{
        int abc,x,z;
        ...
}
func (...)
{
        int z;
        ...
}
main ()
{
   int x,y;
   if (...)
   {
      float A,B,C,x;
      ...
    }
   ...
}
```

Funções

- Parâmetros formais

Os parâmetros formais são declarados como sendo as entradas de uma função. Um parâmetro formal é uma variável local da função que é inicializada com valores externos à função.

Ao se alterar o valor de um parâmetro formal esta alteração não terá efeito na variável que foi passada à função. Isto ocorre pois, em C, quando se passa parâmetros para uma função, são copiados os valores das variáveis, expressões ou constantes para os parâmetros formais. Isto é, os parâmetros formais existem independentemente das variáveis que foram passadas para a função, estes apenas são inicializados com uma cópia dos valores passados para a função.

Funções

- Parâmetros formais (continuação)

Exemplo:

```
#include <stdio.h>
void f1(int i)
{
   i=18;
}
main()
{
   int i=3;
   f1(i);
   printf (″%d″,i);
   return 0;
}
```

Funções

Exercício:

Escreva o código fonte de um programa, na linguagem C, que manipule um vetor de inteiros com dez elementos. O programa deve possuir uma função que receba o vetor e retorne o maior valor contido no mesmo. As seguintes manipulações devem ser feitas: o vetor deve ser inicializado e, por meio da utilização da função mencionada, o maior valor contido no vetor deve ser impresso na saída padrão.

```
#include <stdio.h>
#define n 10
int maior_valor (int vetor[n])
{
   int i,maior_v;
   for(i=0;i<n;++i)
      if(!i)
         maior_v=vetor[i];
      else
         if(maior_v<vetor[i])
            maior_v=vetor[i];
   return maior_v;
}
main()
{
   int i,v[n];
   for(i='\0';i<n;++i)
   {
      printf("\nEntre com o elemento v[%d]: ",i+1);
      scanf("%d",&v[i]);
   }
   printf("\nO maior valor contido no vetor eh: %d",maior_valor(v));
   return 0;
}
```

```
#include <stdio.h>
#define n 10
int maior_valor (int vetor[n])
{
   int i,maior_v;
   maior_v=vetor[0];
   for(i=1;i<n;++i)
      if(maior_v<vetor[i])
         maior_v=vetor[i];
   return maior_v;
}
main()
{
   int i,v[n];
   for(i='\0';i<n;++i)
   {
      printf("\nEntre com o elemento v[%d]: ",i+1);
      scanf("%d",&v[i]);
   }
   printf("\nO maior valor contido no vetor eh: %d",maior_valor(v));
   return 0;
}
```

```
#include <stdio.h>
#define n 2
void teste (int vetor[n]) {
   int aux;
   aux = vetor[0];
   vetor[0] = vetor[1];
   vetor[1] = aux;
}
main() {
   int i,v[n];
   for(i='\0'; i<n; ++i) {
      printf("\nEntre com o elemento v[%d]: ",i+1);
      scanf("%d",&v[i]);
   }
   printf("\nO vetor fornecido eh: \nVetor = {");
   for(i='\0'; i<n; ++i)
      printf(" %d,", v[i]);
   printf ("\b }");
   teste (v);
   printf("\nO vetor apos a chamda da funcao eh: \nVetor = {");
   for(i='\0'; i<n; ++i)
      printf(" %d,", v[i]);
   printf ("\b }");
   return 0;
}
```

No caso de parâmetros formais que são vetores ao serem alterados dentro da função estas alterações se refletem fora!

Funções

- Variáveis globais

Variáveis globais são declaradas fora de todas as funções do programa. Elas são conhecidas e podem ser alteradas por todas as funções do programa que sucedem sua declaração. Quando uma função tem uma variável local com o mesmo nome de uma variável global a função dará preferência à variável local. Vamos ver um exemplo:

```
func1 (...)
{
        int x,y;
        ...
}
int z,k;
func2 (...)
{
        int x,y,z;
        ...
        z=10;
        ...
}
main ()
{
        int count;
        z=7;
        func2(...);
        ... /* z?*/
}
```

Exercício:

Analise o seguinte programa e indique o que será impresso na saída padrão.

```
#include <stdio.h>
int num;
int func(int first, int sec)
{
   first = (first+sec)/2;
   num -= first+1;
   return first;
}
 main()
{
   int first = 0, sec = 50;
   num = 10;
   printf("\nnum antes = %d\tfirst antes = %d\tsec antes = %d", num,
first, sec);
   num += func(first, sec);
   printf("\nnum depois = %d\tfirst depois = %d\tsec depois = %d",
num, first, sec);
}
```

num antes = 10

num depois = 9

first antes = 0

first antes = 0

sec antes = 50

sec antes = 50

Funções

Exercício:

Escreva o código fonte de um programa, na linguagem C, que manipule um vetor de inteiros com dez elementos. O programa deve possuir uma função responsável pela inicialização do vetor e outra função que efetue a impressão do vetor na saída padrão com um layout adequado. Por meio das funções mencionadas, o programa deve inicializar o vetor e em seguida retorná-lo no monitor.

```
#include <stdio.h>
#define n 10
int vetor[n];
void inicializar_vetor()
{
   int i;
   for(i='\0';i<n;++i)
   {
      printf("\nEntre com o elemento v[%d]: ",i+1);
      scanf("%d",&vetor[i]);
   }
}
```

```
void imprimir_vetor()
{
   int i;
   for(i='\0';i<n;++i)
      if(!i)
         printf("vetor = { %d, ", vetor[i]);
      else
         if(i==n-1)
            printf("%d }", vetor[i]);
         else
            printf("%d, ", vetor[i]);
}
```

```
int main()
{
   inicializar_vetor();
   imprimir_vetor();
   return 0;
}
```