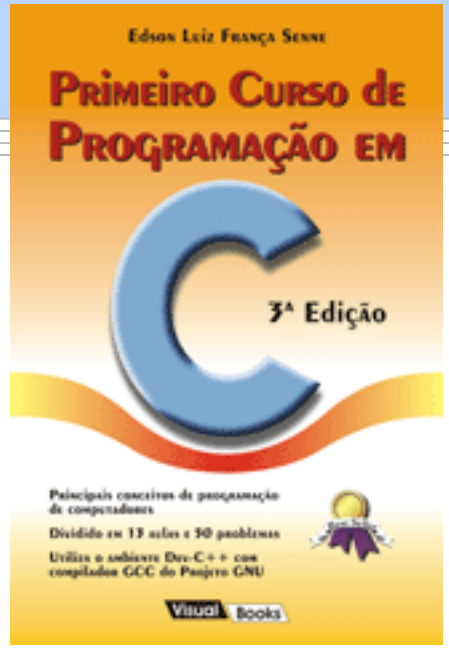
Linguagem C



Estrutura de um programa C:

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

int main(int argc, char *argv[])
{
   float y;
   y = sin(1.5);
   printf("y = %f",y);
   printf("\n");
   system("PAUSE");
   return 0;
}
```

Linguagem C

Outro exemplo:

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

int main(int argc, char *argv[])
{
  float d,p,s,t;

  d = 500;   // depósito inicial
  // após o primeiro mês
  p = d + 0.01*d;
  // após o segundo mês
  s = p + 0.01*p;
  // após o terceiro mês
  t = s + 0.01*s;
  printf("Valor da conta = %.2f\n",t);
  system("pause");
  return 0;
}
```

Linguagem C

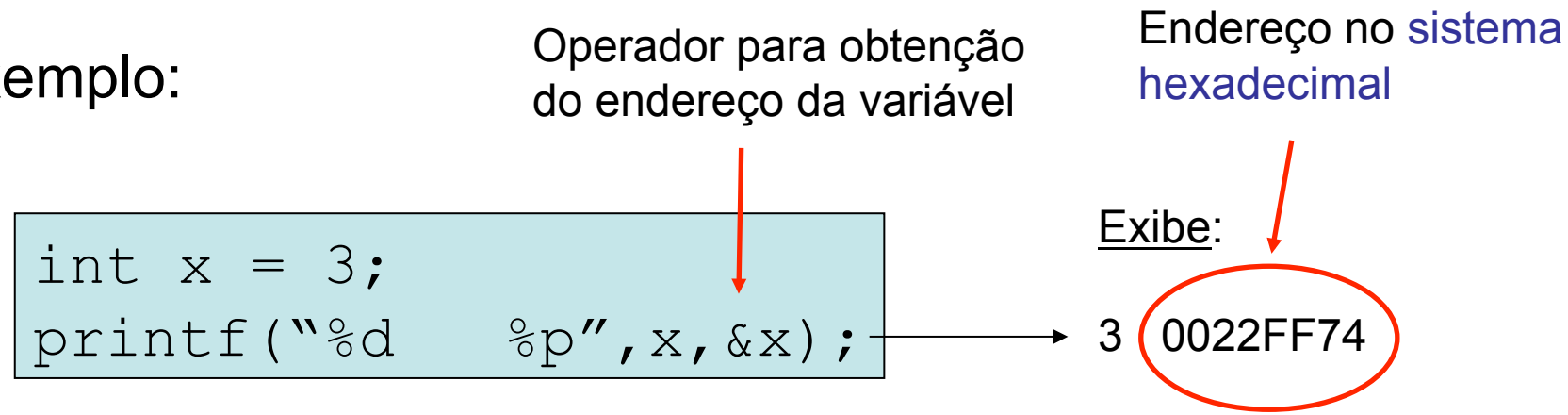
Exemplo 3:

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

int main(int argc, char *argv[])
{
  int x;
  short int y;
  char a;
  unsigned char b;

  x = pow(2,31)-1;      // maior int possível
  y = pow(2,15)-1;      // maior short int possível
  printf("x = %d  y = %d\n",x,y);
  x = x + 1;
  y = y + 1;
  printf("x = %d  y = %d\n",x,y);
  /* ------------
     Atribuir os maiores valores possíveis
     para as variáveis a e b.
     ------------------------------------ */
  a = pow(2,7)-1;
  b = pow(2,8)-1;
  printf("a = %d  b = %d\n",a,b);
  a = a + 1;
  b = b + 1;
  printf("a = %d  b = %d\n",a,b);
  system("PAUSE");
  return 0;
}
```

Endereços de Variáveis

- Uma variável representa um nome simbólico para uma posição de memória.
- Cada posição de memória de um computador possui um endereço. Logo, o endereço de uma variável é o endereço da posição de memória representada pela variável.

Exemplo:



Endereços de Variáveis

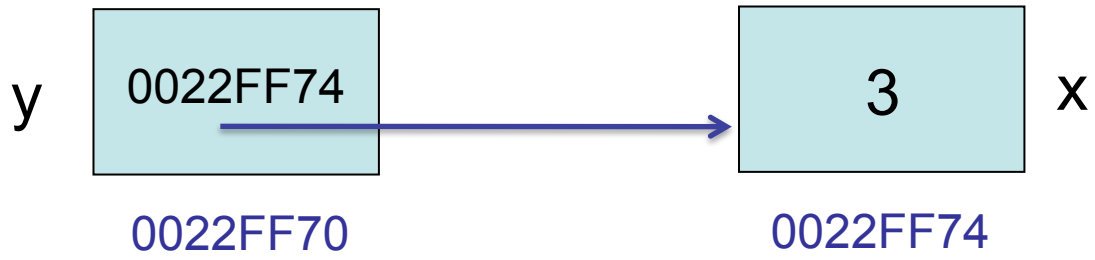
- Note que o endereço de uma variável é um valor. Logo, uma variável pode armazenar um endereço.

- Uma variável que armazena um endereço de memória é conhecida como ponteiro (*pointer*).

- Daí o porque do *tag* usado para exibir endereços de memória ser %p.

Endereços de Variáveis

- Suponha que y armazene o endereço 0022FF74 de uma posição de memória representada pela variável x. E ainda, que x contenha o valor inteiro 3.
- Esquematicamente, podemos representar:

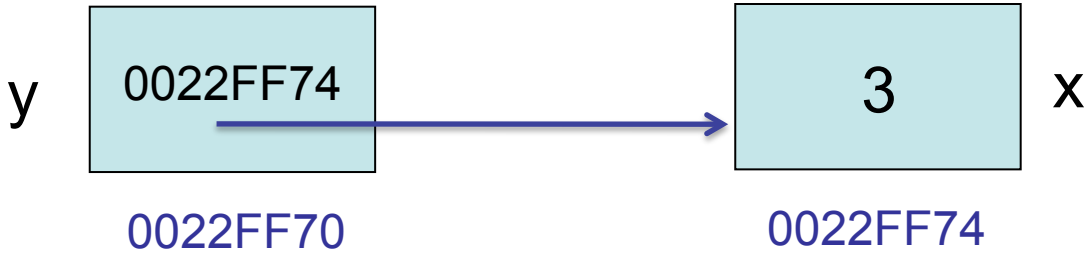


- Dizemos que:

  y é um ponteiro para x, ou

  y aponta para x.

Linguagem C



Qual é o **tipo da variável** y?

- Para declarar um ponteiro é preciso saber **para qual tipo de valor** este ponteiro irá apontar.
- No caso acima, o ponteiro **aponta para um valor inteiro**. Assim, diz-se que o tipo de y é int *.
- A declaração da variável y será:

```
int *y;
```

ou seja: y é um ponteiro para int.

Linguagem C

Exemplo 4:

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

int main(int argc, char *argv[])
{
  int a = 2, b = 3, c = 1;
  float delta,x1,x2;

  delta = b*b - 4*a*c;
  printf("A equacao %s\n", (delta >=0) ? "possui raizes reais" :
                                         "nao possui raizes reais");
  if (delta >= 0)
  {
    printf("As raizes sao %s\n", (delta > 0)? "diferentes" : "iguais");
    x1 = (-b + sqrt(delta))/(2*a);
    x2 = (-b - sqrt(delta))/(2*a);
    printf("Raiz x1 = %f\n", x1);
    printf("Raiz x2 = %f\n", x2);
  }

  system("PAUSE");
  return 0;
}
```

Linguagem C

Exemplo 5: Entrada de dados

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

int main(int argc, char *argv[])
{
  float C,F;
  printf("Digite a temperatura em graus C: ");
  scanf("%f", &C);
  F = (9 * C + 160) / 5;
  printf("Esta temperatura corresponde a %.1f graus F\n", F);
  system("PAUSE");
  return 0;
}
```

Linguagem C

Exemplo 6:

- Dadas as idades (tipo int) e os pesos (tipo float) de 2 pessoas, exibir quem é a pessoa mais velha (e a sua idade) e quem é a pessoa mais leve (e o seu peso).

```c
// Programa p08.c
#include <stdio.h>
#include <stdlib.h>

int maiorValor(int idade1, int idade2)
{
  if (idade1 > idade2)
    return idade1;
  else
    return idade2;
}

int maiorQuem(int idade1, int idade2)
{
  if (idade1 > idade2)
    return 1;
  else
    return 2;
}

float menorValor(float peso1, float peso2)
{
  if (peso1 < peso2)
    return peso1;
  else
    return peso2;
}
```

Linguagem C

```c
int menorQuem(float peso1, float peso2)
{
  if (peso1 < peso2)
    return 1;
  else
    return 2;
}
```

Lembrar que, quando se executa um programa, executa-se a função **main**.

```c
int main(int argc, char *argv[])
{
  int idade1,idade2,maior_idade;
  int mais_velho,mais_leve;
  float peso1,peso2,menor_peso;
  printf("Digite a idade e o peso da pessoa 1: ");
  scanf("%d %f", &idade1,&peso1);
  printf("Digite a idade e o peso da pessoa 2: ");
  scanf("%d %f", &idade2,&peso2);
  maior_idade = maiorValor(idade1,idade2);
  mais_velho = maiorQuem(idade1,idade2);
  menor_peso = menorValor(peso1,peso2);
  mais_leve = menorQuem(peso1,peso2);
  printf("Maior idade = %d (da pessoa %d)\n",maior_idade,mais_velho);
  printf("Menor peso = %.1f (da pessoa %d)\n",menor_peso,mais_leve);
  system("PAUSE");
  return 0;
}
```

Linguagem C

- Observe a função maiorValor:

```
int maiorValor(int idade1, int idade2)
{
   if (idade1 > idade2)
      return idade1;
   else
      return idade2;
}
```

- Esta função funciona apenas para idades?

- Esta função poderá ser usada sempre que for necessário determinar o **maior** de **dois** valores **inteiros**!

Linguagem C

- Podemos escrever a função maiorValor como:

```
int maiorValor(int x, int y)
{
   if (x > y)
      return x;
   else
      return y;
}
```

- Os nomes dos parâmetros **na definição** da função não precisam ser iguais aos nomes dos parâmetros **no uso** da função:

```
m = maiorValor(id1,id2);
```

Linguagem C

Exemplo 7

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

int main(int argc, char *argv[])
{
  int N;
  int i, s;

  printf("Digite o valor de N: ");
  scanf("%d", &N);
  s = 0;
  i = 1;
  while (i <= N)
  {
    s = s + i;
    i++;
  }
  printf("Soma = %d\n", s);
  system("PAUSE");
  return 0;
}
```

Linguagem C

- Outra forma de repetir um conjunto de instruções é com o comando do-while.

```
s = 0;
i = 1;
do
{
   s = s + i;
   i++;
}
while (i <= N);
```

Comando do-while

```
s = 0;
i = 1;
while (i <= N)
{
   s = s + i;
   i++;
}
```

Comando while

- Veja que no comando while, a condição de repetição é testada antes da execução das instruções, ao contrário do comando do-while.
- No exemplo acima, o que acontece para N=0?

Linguagem C

- Na execução de uma repetição pode ser necessário:
  - Encerrar a repetição independentemente do valor da condição do laço;
  - Executar apenas parcialmente uma iteração, ou seja, executar somente algumas das instruções do laço da repetição.

- Para encerrar: comando break.
- Para executar somente algumas das instruções do laço, mas sem encerrar a repetição: comando continue.

Linguagem C

Exemplos:

```
i = 1;
s = 0;
while (i <= N)
{
  s = s + 1.0/i;
  if (s > A)
  {
    printf("Numero de termos = %d\n", i);
    break;
  }
  i++;
}
```

```
printf("Valor de N: ");
scanf("%d", &N);
i = 0;
s = 0;
while (i < N)
{
  i++;
  printf("Idade e peso da pessoa %d: ",i);
  scanf("%d %f",&idade,&peso);
  if (idade <= 30)
    continue;
  s = s + peso;
}
printf("Peso total = %.2f\n",s);
```

Linguagem C

```c
// Programa p14.c
#include <stdio.h>
#include <stdlib.h>

int main(int args, char * arg[])
{
  int i,n;
  char optei;
  char texto[100];

  do
  {
    system("CLS");
    printf("Entre com o texto a ser codificado:\n");
    for (i = 0; i < 100; i++)
    {
      scanf("%c",&texto[i]);
      if (texto[i] == '.')
        break;
    }
    n = i;
    printf("Texto com %d caracteres.\n\n", n);
    for (i = 0; i < n; i++)
    {
      texto[i] = (5*texto[i] + 100) % 256;
    }
```

Linguagem C

```
    printf("Texto codificado:\n");
    for (i = 0; i < n; i++)
      printf("%c",texto[i]);
    printf("\n\n");
    printf("Continua? (S/N): ");
    optei = toupper(getche());
  }
  while (optei == 'S');
  printf("\n");
  system("PAUSE");
  return 0;
}
```

Linguagem C

- Para vetores é muito conveniente usar o comando for.
- Os comandos for e while são equivalentes.

```
for (i = 0; i < 100; i++)
{
   scanf("%c",&texto[i]);
   if (texto[i] == '.')
     break;
}
```

```
i = 0;
while (i < 100)
{
    scanf("%c",&texto[i]);
    if (texto[i] == '.')
       break;
    i++;
}
```