Eduardo Augusto Bezerra <bezerra@eel.ufsc.br> - Jul. 2002

(para maiores detalhes ver: "Practical C++, Rob McGregor" e "C++ Manual de Referencia Comentado, Bjarne Stroustrup")

PONTEIROS EM C++

- Um ponteiro e' um tipo especial de variavel que armazena um endereco de memoria.
- Um ponteiro e' o endereco de um dado (objeto) de um tipo especifico.
- Endereco e' o inicio da area de memoria que contem o dado.
- O tipo determina o tamanho do dado, funcoes e operadores.

```
Obs. Tamanho do dado:

Ex:

char grade;    // 1 byte
short score;   // 2 bytes
float average; // 4 bytes

char *pGrade = &grade;     // 4 bytes (tamanho de um inteiro)
short *pScore = &score;     // 4 bytes (tamanho de um inteiro)
float *pAverage = &average; // 4 bytes (tamanho de um inteiro)

Fim-Obs.

variavel ponteiro p

  p          *p
+----+     +----+
|    |---->| 25 |
+----+     +----+

cout << *p; // escreve 25
*p = 30;    // altera *p
```

- O operador unario * possibilita que um dado seja referenciado indiretamente.

Declaracao de uma variavel do tipo ponteiro:

```
int *p; // p e' um ponteiro para inteiros
double *q;

em C++

int* p;
double* q;

Obs. Para declarar 3 ponteiros na mesma linha:

int *p, *q, *r;

Fim-Obs.
```

- O endereco de um ponteiro e' desconhecido antes da compilacao/execucao.
- Utilizar o operador unario & em uma variavel. Esse operador retorna um ponteiro para o objeto.

```
Ex: int m = 0;
    int n = 0;
    int* p = &n; // variavel ponteiro p aponta para variavel inteiro n
    // *p e' um alias (nome alternativo) de n
    *p = 30; // altera n
    p = &m; // agora *p aponta para m e nao n
    *p = 20; // altera m
```

EX:

```
#include <iostream>
using namespace std;
void main() {
   int var = 50;
   int *pVar;
   pVar = &var;
   cout << "Variavel var apontada por pVar == " << *pVar
   << "\nvar esta no endereco " << pVar << endl;
}
```

Obs. **IMPORTANTE!!!!**

Entender a diferenca entre *pVar e var, e tambem entre *p, m e n

Fim-Obs.

Ponteiros para "structs"

```
#include <iostream>
using namespace std;

typedef struct {
   int largura;
   int altura;
   unsigned *bits;
} Image;

void main() {
   Image pic1;
   Image *pImage = &pic1;
   pic1.largura = 100;

   cout << "pic1.largura == " << pic1.largura << endl;
   cout << "pImage->largura == " << pImage->largura << endl;
   cout << "(*pImage).largura == " << (*pImage).largura << endl;
   cout << "(&pic1)->largura == " << (&pic1)->largura << endl;

   // -> e' o operador ponteiro para membro de struct, class, union

   // Quais as saidas apos a execucao da instrucao abaixo?

   (*pImage).largura = 50;

   cout << "pic1.largura == " << pic1.largura << endl;
   cout << "pImage->largura == " << pImage->largura << endl;
   cout << "(*pImage).largura == " << (*pImage).largura << endl;
```

```
   cout << "(&pic1)->largura == " << (&pic1)->largura << endl;
}

Ex:

struct date {
   int day;
   int month;
   int year;
};
date dt;

date* pDt = &dt;

// . tem precedencia sobre * USAR PARENTESES!

dt.day = 14;     (*pDt).day = 14;       pDt->day = 14;
dt.month = 8;    (*pDt).month = 8;      pDt->month = 8;
dt.year = 2002; (*pDt).year = 2002;    pDt->year = 2002;
```

Uso de ponteiros para passagem de parametros para funcoes

```
Ex: struct poligono {
        int cor_da_borda;
        int cor;
        int pt[1000];
};
```

- Estrutura imaginaria representando um poligono de ate' 1000 vertices.
- Enviar essa estrutura para uma funcao significa gasto em tempo de processamento e memoria para armazenamento -> copiar para a pilha 4008 bytes (1 inteiro ==- 4 bytes), e no final da execucao desempilhar tudo.
- O prototipo da funcao seria:

```
void f1(poligono meuPoligono){
   ...
   meuPoligono.cor_da_borda = 1;
   ...
}
```

- Passando o ponteiro, o prototipo da funcao seria:

```
void f1(poligono *meuPoligono){
   ...
   meuPoligono->cor_da_borda = 1;
   ...
}

Ex: Escrever a funcao swap(i, j) e a chamada para essa funcao. A funcao recebe duas
variaveis inteiras e realiza a troca dos conteudos dessasvariaveis entre elas. Apos a
execucao, i contera' j e j contera' i.

Solucao: Passar os enderecos das variaveis na memoria.

void swap(int* i, int* j) {
   int tmp = *i;
   *i = *j;
```

```
    *j = tmp;
}

//chamada:
swap(&x, &y);
```

Arrays e ponteiros

- Considernado as definicoes "int a[100], b[100];", em qualquer expressao, "a" se comportara' como &a[0], e "b" como &b[0]

Ex:

```
int M[100];
int *pM;
pM = &M[0]; // ou pM = M
```

Ex: Definir a funcao soma(M, n) que retorna o somatorio dos n primeiros elementos do array M.

```
int soma(int* p, int n){
   int total = 0;
   for (int i = 0; i < n; i++)
      total += p[i];
   return total;
}

// chamada:
soma (M, 50); // soma dos 50 primeiros elementos de M
```

```
  p          M
+----+     +----+----+----+----+----+----+----+
|    |---->| 25 | -2 | 12 | 62 | 73 | 33 | 12 | ...
+----+     +----+----+----+----+----+----+----+
```

```
M[0] == p[0] == *p
M[1] == p[1] == *(p+1)
```

Ex: Escrever funcao para copiar n elementos do array q para o array p.

```
void copy(int* p, int* q, int n) {
   for (int i = 0; i < n; i++)
      p[i] = q[i];
}

// chamada:
copy (a, b, 75);
```

Aritmetica de ponteiros

- Os operadores ++ e -- podem ser utilizados para mover ponteiros na memoria.
- O numero de bytes sobre os quais o ponteiro se desloca depende do tipo da variavel ponteiro em questao.
- Por exemplo, em um ponteiro do tipo char, um incremento na variavel ponteiro representao o salto na memoria para o byte seguinte.
- Ja' em um ponteiro do tipo int, o salto sera' de sizeof(int) bytes.

```
Ex: int* p = a;

// p aponta para o elemento 0 do array A (*p == A[0])
// ao se executar p = p + 1; fara' p apontar para A[1]

Ex: x = *p++; // x recebe o valor apontado por p, e apos isso o ponteiro e' incrementado.

x = *(p++); // x recebe o valor apontado por p, e apos isso o ponteiro e' incrementado.

Ex: x = *++p; // p e' incrementado, e apos o valor e' utilizado.

x = *(++p); // ponteiro e' incrementado, e o conteudo da nova posicao e' atribuido a x.

Ex:

void copy(int* p, int* q, int n) {
   for (int i = 0; i < n; i++)
      *p++ = *q++;
}

Ex:

int sum (int* p, int n){
   int total = 0;
   for (int i = 0; i < n; i++)
      total += *p++;
   return total;
}

Ex: Programa que utiliza aritmetica de ponteiros para mostrar uma string na ordem certa e
na ordem reversa.

#include <string.h>
#include <iostream>
using namespace std;

char * ReverseString(char *pStr){
   int len = strlen(pStr); // numero de caracteres
   char *s = &(pStr[0]); // s aponta para o inicio da string
   char *e = &(pStr[len-1]); // e aponta para o final da string
   char temp;

   do {
      // troca caracteres das extremidades
      temp = *s;
      *s = *e;
      *e = temp;
      // move para dentro da string a partir das extremidades
      ++s;
      --e;
   } while (s < e); // repete ate' a metade da string
   return pStr;
}

void main() {
  char str[30];
  strcpy(str, "Esta eh uma string de teste.");
  char *ch = &str[0];
   for (unsigned i = 0; i < strlen(str); i++)
      cout << *ch++;
   cout << endl << endl << ReverseString(str) << endl;
```

errado.

total = total + *p++

```
}
```

A funcao abaixo pode ser utilizada como base para escrever uma funcao para leitura controlada do teclado.

```
/*****************************************************************/
/* byte Ler (int col, int lin, char *str, int tamanho, int flag, flag1)      */
/* Eduardo Bezerra, Janeiro de 1992                                          */
/*****************************************************************/
byte Ler(int col, int lin, char *str, int tamanho, int flag,int flag1){
  int i, c, flcaneta = TRUE;
  char inf, sup;

  inf = flag ? ' ' : '0';
  sup = flag ? '}' : '9';
  gotoxy(col,lin);

#ifndef EMULA
  if (flag1)
  do{
    caneta ();
    if (*bufcta){
      putch (BELL);
      StrSub (str,bufcta,5,8);
      str[9] = '\0';
      str[0] = '0';
      flcaneta = FALSE;
    }
    c = bdos(0x06,0xff);
  } while ( !c && flcaneta );
#endif

if (flcaneta){
  i = 0;
  do{
    if (!flag1) /* Se flag1 == .T. , primeiro caracter foi */
      c = getch(); /* lido por caneta(), os demais pelo tecl. */
    /* Se flag1 == .F. , todos pelo teclado */
    flag1 = FALSE;
    c = toupper(c);
    if ((c == 'X') && !flag)
      break;
    switch (c){
      case BKSPC : if(i == 0)
                     i--;
                   else{
                     putch(BKSPC); putch(' '); putch(BKSPC);
                     i -= 2;
                    *str-- = '\0';
                   }
                   break;

      default :
                   if (c >= inf && c <= sup){
                     if (i < tamanho){
```

```
                        putch(c);
                        *str++ = c;
                    }
                    else i--;
                }
                else i--;
        }
        i++;
    } while (c != ENTER);
    *str = '\0';
  }
  else
    c = ENTER;
  return c == ENTER ? OK : c;
  }
```