Gestão de Centrais

Formação	**Jogo do Rato**
Local	Marinha Grande
Formador	Sérgio Lopes, knitter.is@gmail.com [mailto:knitter.is@gmail.com]
Ficha	**1 - Ponteiros e Gestão de Centrais**

Ponteiros

Um ponteiro é uma variável, que tal como outras variáveis, reside em memória, possui um endereço e um valor. No entanto os ponteiros guardam apenas um tipo especial de valores: endereços de memória de outras variáveis.

Quando uma variável é declarada, o sistema operativo trata de reservar um espaço de memória com o tamanho necessário e de devolver ao programa o endereço da zona de memória reservada. É com esse endereço que todos os acessos são feitos, sem que o programador tenha de conhecer ou manipular endereços de memória, e através do uso da variável o programador está a usar indirectamente o endereço de memória. Em C é possível manipular directamente o acesso à memória através do uso de ponteiros.

Figura 1. Variáveis Tradicionais

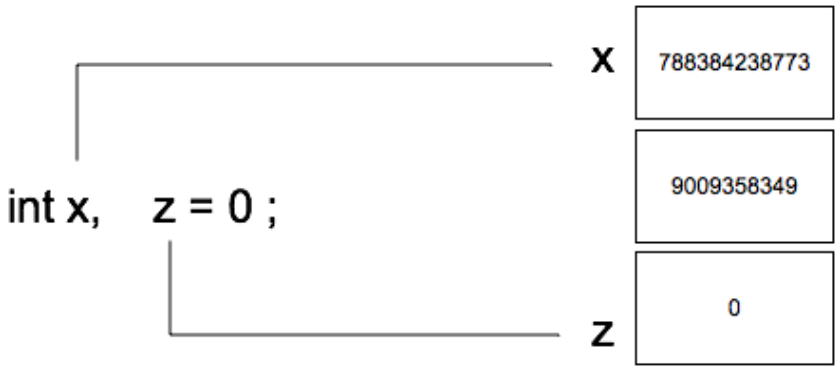
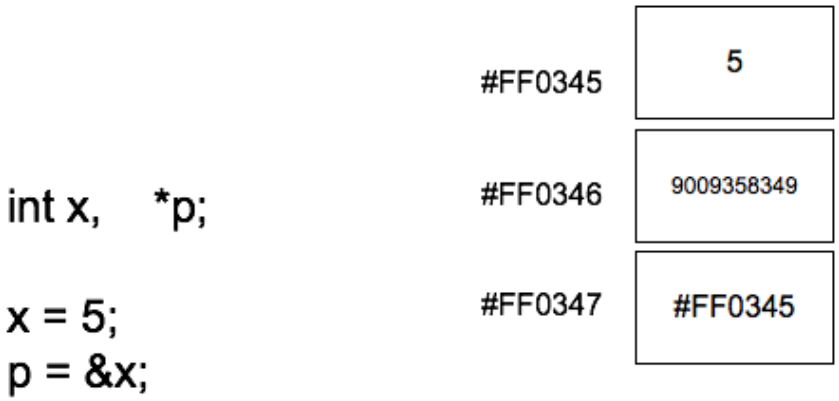


Figura 2. Ponteiros na Memória



A sintaxe de ponteiros faz uso de dois caracteres principais: ***** e **&**. O primeiro permite aceder ao valor da variável para a qual o ponteiro aponta, o segundo é usado para aceder ao endereço de uma variável e assim atribuir o valor desse endereço a um ponteiro.

Tipos de Ponteiros

Os ponteiros possuem sempre um tipo de dados associado, e que controla as operações aplicadas aos ponteiros. Se um ponteiro é declarado como *ponteiro de inteiros* então não poderá apontar para um endereço de uma variável que seja declarada como *float*. A única excepção são os ponteiros para o tipo de dados *void*, dado que este tipo de dados é especial, os ponteiros para *void* podem depois ser usados para apontar para qualquer outra variável, no entanto cabe ao programador ter o cuidado de garantir que todos os acessos são válidos.

Ponteiros e funções

Além de variáveis, a memória de um programa guarda também as funções que estão carregadas e que pertencem a esse programa. Seguindo essa característica, e considerando que um ponteiro guarda apenas endereços de memória, é possível criar um ponteiro que aponte para um endereço de memória de uma função. Esta característica será vista mais tarde.

Regras

A Um ponteiro guarda apenas um endereço, não guarda valores numéricos, letras, ou qualquer outro tipo de dados, apenas um endereço de memória.

B Um ponteiro é dependente do tipo de dados para o qual aponta. Um ponteiro para *char* só pode guardar caracteres e um ponteiro para *int* só pode guardar inteiros.

C Como qualquer outra variável, um ponteiro tem um valor e um endereço de memória mas acrescenta a capacidade de referenciar outros valores das variáveis para as quais aponta.

D A sintaxe para acesso a ponteiros é confusa! Devem ter atenção ao código escrito e fazer uso de parêntesis sempre que se misturem ponteiros com outras operações, quando não têm a certeza da prioridade dos operadores ou quando se usem formas de ponteiros que não são padrão, ex:

```
int y = 9, *p, b;
p = &y;

b = *p++; // b fica com valor 9,
          //p aponta para zona de memória seguinte

b = *(p++); //igual ao anterior

b = (*p)++; //b fica com o valor 9
            //incrementa b uma unidade, b fica com 10

b = *++p; //p é incrementado uma unidade, p e não o valor!
          //p fica assim para a zona de memória seguinte
          //b fica com valor imprevisível

b = ++*p; //incrementa y, coloca o valor de y em b
          //b fica com valor 10
```

E Um vector é um ponteiro para a primeira posição de um conjunto de memória sequencial. No entanto é um ponteiro constante, sem que possa ser alterado o seu valor inicial.

F O uso de ponteiros é perigoso! A utilização de ponteiros permite a manipulação directa da memória. O compilador não consegue validar o nosso código e garantir que o que estamos a fazer é válido. Do mesmo modo, o sistema operativo não controla o que o programa faz e é fácil aceder a zonas de memória que não nos pertencem e danificar outros programas instalados e a correr ao mesmo tempo, ou até inutilizar um sistema operativo.

Os sistemas actuais possuem mecanismos que permitem detectar alguns comportamentos incorrectos e forçar o programa a terminar, no entanto é necessário ter atenção ao código que é feito.

Vectores

Como está indicado nas regras acima, um vector é nada mais que um ponteiro para a primeira posição de um conjunto de zonas de memória seguidas. Este ponteiro é declarado como constante, e por isso nunca podemos mudar o seu valor inicial, mas podemos aceder-lhe com a aritmética de ponteiros (somando valores à posição inicial para obter as posições seguintes).

Ponteiros e Passagem por Referência

Como temos visto, as funções apenas podem devolver um valor, não é possível devolver mais que um, e mesmo que apenas pretendamos devolver um, por vezes isso não é possível (tentem devolver um vector declarado dentro de uma função e vejam os resultados). Para contornar este problema, e porque a passagem de parâmetros para uma função e a devolução de valores é feita por cópia causando um aumento no consumo de memória, usamos ponteiros para referenciar variáveis e, em vez de passarmos a variável, passamos o ponteiro para ela.

Exemplo de Passagem por Referência.

```c
void dobro(int *);

int main(int argc, int **argv) {

    int x, *p;

    p = &x;

    printf("Insira um valor inteiro: ");
    scanf("%d", &x);

    dobro(p);

    printf("\nDobro: %d\n\n", x);

    return 0;
}

void dobro(int *a) {
    *a = 2 * (*a);
}
```

Gestão de Centrais

Fazendo uso de ponteiros, incluindo a passagem de valores por referência, implemente o projecto descrito abaixo.

Programa que permita guardar a produção de energia eléctrica mensal (em KW) para todo um ano e para cada central existente, no máximo 100 centrais diferentes e todas pertencem ao mesmo ano. É assim necessário guardar as seguintes informações:

a. Código da Central (Alfanumérico)

b. Nome da Central

c. Coordenadas GPS (Latitude e Longitude, alfanumérico)

d. Tipo de Central (Eólica ou Fotovoltaica)

e. Custo do parque

f. Valores de energia produzidos para cada um dos 12 meses

O programa deverá apresentar o seguinte menu de opções:

1. Gestão de Centrais

2. Leituras

3. Estatísticas

4. Gravar informação

5. Ler informação de ficheiro

6. Sair

Para cada uma das opções anteriores deve ser criada um menu que permita efectuar a correcta gestão da informação tendo em conta que o programa deve oferecer as seguintes funcionalidades:

a. Gestão de centrais (inserir, consultar por nome, consultar por tipo, ver registos de energia para o ano, ver localização de todas as centrais, apagar centrais);

b. Gestão e leituras (inserir leituras de energia para uma central e para um mês, inserir todas as leituras do ano para determinada central, substituir uma leitura;

c. Estatísticas sobre central mais produtiva, central menos produtiva, tipo de energia mais produtivo, tipo de energia menos produtivo, média anual de uma central, mês mais produtivo;

d. Gravar dados em ficheiro binário com caminho a indicar pelo utilizador;

e. Ler dados de um ficheiro previamente gravado, com caminho a indicar pelo utilizador;