LABORATÓRIO DE ALGORITMOS E ESTRUTURAS DE DADOS I

Prof. Caio César de Freitas Dantas

Ponteiros – Operações Aritméticas

- Podemos fazer algumas operações aritméticas com ponteiros.
- A operação mais simples, é igualar dois ponteiros. Se temos dois ponteiros p1 e p2 podemos igualá-los fazendo p1=p2.
- Observe que estamos fazendo p1 apontar para o mesmo lugar que p2.
- Se quisermos que a variável apontada por p1 tenha o mesmo conteúdo da variável apontada por p2 devemos fazer *p1=*p2.
- Depois que se aprende a usar os dois operadores (& e *) fica fácil entender operações com ponteiros.

Ponteiros – Operações Aritméticas

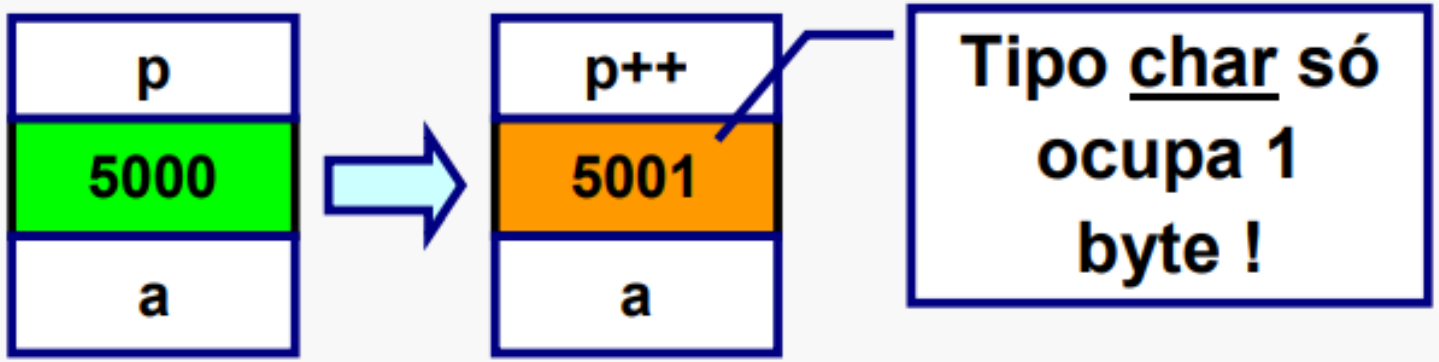
- Um tipo de operação muito usada, é o incremento e o decremento (por exemplo: p+1 e p-1).
- Quando incrementamos um ponteiro p ele passa a apontar para o próximo valor do mesmo tipo para o qual o ponteiro aponta. Isto é, se temos um ponteiro para um inteiro e o incrementamos ele passa a apontar para o próximo inteiro.
- Esta é mais uma razão pela qual o compilador precisa saber o tipo de um ponteiro: se você incrementa um ponteiro *char ele anda 1 byte na memória e se você incrementa um ponteiro *double ele anda 8 bytes na memória.
- O decremento funciona semelhantemente. Supondo que p é um ponteiro, as operações podem ser escritas como:

```
p++;    p--;
```

Ponteiros – Operações Aritméticas

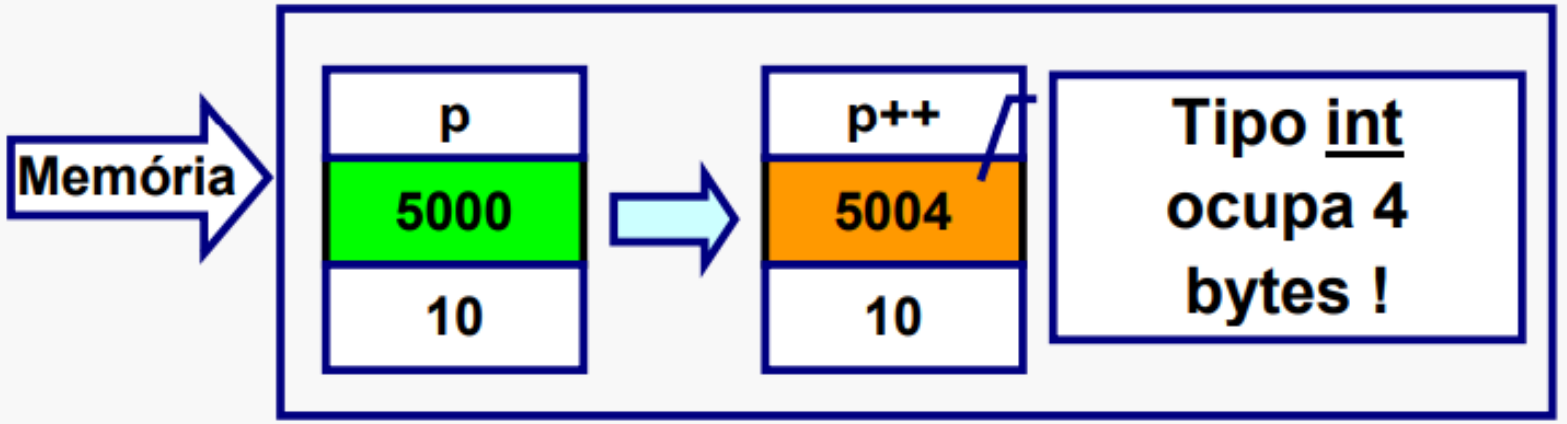
- Uma variável do tipo ponteiro está sempre associada a um tipo. Ou seja, um ponteiro para um dado tipo t endereça o número de bytes que esse tipo t ocupa em memória

```
char ch, *p;
ch = 'a';
p = &ch;
p++;
```



Ponteiros – Operações Aritméticas

```
int x, *p;
x = 10;
p = &x;
p++;
```



Espaço de 4 bytes para armazenar a variável x tipo int.

A operação p++ percorre sizeof(tipo p) bytes !

p				p++			
x	x	x	x	?	?	?	?
5000	5001	5002	5003	5004	5005	5006	5007

Ponteiros – Declarando e Utilizando

- Exemplo 4

```
main()
{ int *px1;
 float *px2;
 char *px4;
 double *px3;
 int i=1;
 float f= 0.3;
 double d= 0.005;
 char c = '*';
 px1=&i;
 px2=&f;
 px3=&d;
 px4=&c;
```

```
printf("Valores ponteiros antes ++:px1=%d px2=%d
     px3=%d px4=%d  \n\n",px1, px2,px3,px4);
```

Ponteiros – Declarando e Utilizando

- Exemplo 4

```
Valores ponteiros antes ++:  px1=2293604  px2=2293600  px3=2293592  px4=2293591
```

```
px1++;
px2++;
px3++;
px4++;
```

```
printf("Valores dos ponteiros depois ++:px1=%d
        px2=%d px3=%d  px4=%d\n\n",px1,px2,px3,px4);
```

```
Valores ponteiros depois ++:px1=2293608  px2=2293604  px3=2293600  px4=2293592
```

Ponteiros – Operações Aritméticas

- Estamos falando **de operações com ponteiros e não de operações com o conteúdo das variáveis** para as quais eles apontam. Por exemplo, para incrementar o conteúdo da variável apontada pelo ponteiro p, faz-se: (*p)++;

- Outras operações aritméticas úteis são a soma e subtração de inteiros com ponteiros. Suponha que você queira incrementar um ponteiro de 15. Basta fazer: p=p+15; ou p+=15;

- Se você quiser usar o conteúdo do ponteiro 15 posições adiante basta fazer *(p+15);

- A subtração funciona da mesma maneira que a adição.

Ponteiros – Operações Aritméticas

Expressões com ponteiros

- Atribuição de ponteiros;
- Aritmética de ponteiros;
- Comparação de ponteiros;

Ponteiros – Operações Aritméticas

Como é o caso com qualquer variável, um ponteiro pode ser usado no lado direito de um comando de atribuição para passar seu valor para outro ponteiro.

```
int main() {
int x, *p1, *p2;
x=10;
p1 = &x;
p2 = p1;
printf("O endereco de p2 e: %d \n\n", p2);
printf("O conteudo apontado por p1 e: %d \n\n", *p1);
printf("O conteudo apontado por p2 e: %d \n\n", *p2);
return 0;
}
```

Ponteiros – Operações Aritméticas

Saída do Programa:

```
O endereco de p2 e: 6487564

O conteudo apontado por p1 e: 10

O conteudo apontado por p2 e: 10
```

Ponteiros – Operações Aritméticas

Aritmética de ponteiros;

- Existem apenas duas operações aritméticas que podem ser usadas com ponteiros:
  - Adição (incremento)
  - Subtração (decremento).

- Quando incrementamos um ponteiro ele passa a apontar para o próximo valor do mesmo tipo para o qual o ponteiro aponta;
  - Se tivermos um ponteiro para um inteiro e o incrementamos ele passa a apontar para o próximo inteiro;

Ponteiros – Operações Aritméticas

Aritmética de ponteiros;

Isso justifica a necessidade do compilador conhecer o tipo de um ponteiro;

- Se incrementarmos um ponteiro char * ele anda 1 byte na memória;
- Se incrementarmos um ponteiro double * ele anda 8 bytes na memória;

Ponteiros – Operações Aritméticas

Aritmética de ponteiros;

```
int main() {
int *p1, x;
x=10;
p1=&x;
printf("O endereco de p1 e: %d \n\n", p1);
p1++;
printf("O endereco de p1 e: %d \n\n", p1);
return 0;
}
```

Ponteiros – Operações Aritméticas

Aritmética de ponteiros;

```
O endereco de p1 e: 6487572

O endereco de p1 e: 6487576
```

```c
int main() {
int *p1, x;
x=10;
p1=&x;
printf("O endereco de p1 e: %d \n\n", p1);
p1++;
printf("O endereco de p1 e: %d \n\n", p1);
return 0;
}
```

Ponteiros – Operações Aritméticas

Aritmética de ponteiros;

- A aritmética de ponteiros não se limita apenas ao incremento e decremento, podemos somar ou subtrair inteiros de ponteiros.

Ponteiros – Operações Aritméticas

Aritmética de ponteiros;

- A aritmética de ponteiros não se limita apenas ao incremento e decremento, podemos somar ou subtrair inteiros de ponteiros.

```
int main() {
int *p1, x;
x=10;
p1=&x;
printf("O endereco de p1 e: %d \n\n", p1);
p1= p1+3;
printf("O novo endereco de p1 e: %d \n\n", p1);
return 0;
}
```

Ponteiros – Operações Aritméticas

Aritmética de ponteiros;

- A aritmética de ponteiros não se limita apenas ao incremento e decremento, podemos somar ou subtrair inteiros de ponteiros.

```c
int main() {
int *p1, x;
x=10;
p1=&x;
printf("O endereco de p1 e: %d \n\n", p1);
p1= p1+3;
printf("O novo endereco de p1 e: %d \n\n", p1);
return 0;
}
```

```
O endereco de p1 e: 6487572

O novo endereco de p1 e: 6487584
```

Ponteiros – Operações Aritméticas

Aritmética de ponteiros;

- Como faríamos para incrementar o conteúdo da variável apontada por um ponteiro p qualquer? (*p)++

Ponteiros – Operações Aritméticas

Aritmética de ponteiros;

- Como faríamos para incrementar o conteúdo da variável apontada por um ponteiro p qualquer? (*p)++

```
int main() {
int *p1, x;
x=10;
p1=&x;
printf("O endereco de p1 e: %d \n\n", p1);
printf("O conteudo de p1 e: %d \n\n", *p1);
(*p1)++;
printf("O novo conteudo de p1 e: %d \n\n", *p1);
return 0;
}
```

Ponteiros – Operações Aritméticas

Aritmética de ponteiros;

- Como faríamos para incrementar o conteúdo da variável apontada por um ponteiro p qualquer? (*p)++

```
O endereco de p1 e: 6487572

O conteudo de p1 e: 10

O novo conteudo de p1 e: 11
```

```c
int main() {
int *p1, x;
x=10;
p1=&x;
printf("O endereco de p1 e: %d \n\n", p1);
printf("O conteudo de p1 e: %d \n\n", *p1);
(*p1)++;
printf("O novo conteudo de p1 e: %d \n\n", *p1);
return 0;
}
```

Ponteiros – Operações Aritméticas

Aritmética de ponteiros;

- Além de adição e subtração entre um ponteiro e um inteiro, nenhuma outra operação aritmética pode ser efetuada com ponteiros;
    - Não podemos multiplicar ou dividir ponteiros;
    - Não podemos adicionar ou subtrair o tipo float ou o tipo double a ponteiros;

Ponteiros – Operações Aritméticas

Comparação de ponteiros

- É possível comparar dois ponteiros em uma expressão relacional (<=, > e >=) ou se eles são iguais ou diferentes (== e !=);

- A comparação entre dois ponteiros se escreve como a comparação entre outras duas variáveis quaisquer.

Ponteiros – Operações Aritméticas

Comparação de ponteiros

```
int main() {
int *p1, *p2;
int x=10, y=10;
p1=&x;
p2=&y;
if (p1 > p2)
    printf("x esta armazenada em um endereco de memoria acima de y \n\n");
else
    printf("y esta armazenada em um endereco de memoria acima de x \n\n");

printf("Mostrando enderecos de x = %d e y = %d  \n\n", p1, p2);
return 0;
}
```

Ponteiros – Operações Aritméticas

Comparação de ponteiros

```
x esta armazenada em um endereco de memoria acima de y

Mostrando enderecos de x = 6487564 e y = 6487560
```

```c
int main() {
int *p1, *p2;
int x=10, y=10;
p1=&x;
p2=&y;
if (p1 > p2)
    printf("x esta armazenada em um endereco de memoria acima de y \n\n");
else
    printf("y esta armazenada em um endereco de memoria acima de x \n\n");

printf("Mostrando enderecos de x = %d e y = %d  \n\n", p1, p2);
return 0;
}
```

Ponteiros – Operações Aritméticas

Escreva um programa que:

- Declare um inteiro, um real e um char, e ponteiros para cada um deles.
- Associe as variáveis aos ponteiros.
- Mostre o endereço de cada variável.
- Atribua valores a cada variável usando os ponteiros.
- Imprima os valores das variáveis.
- Modifique os valores de cada variável usando os ponteiros.
- Imprima os novos valores das variáveis.

FIM!