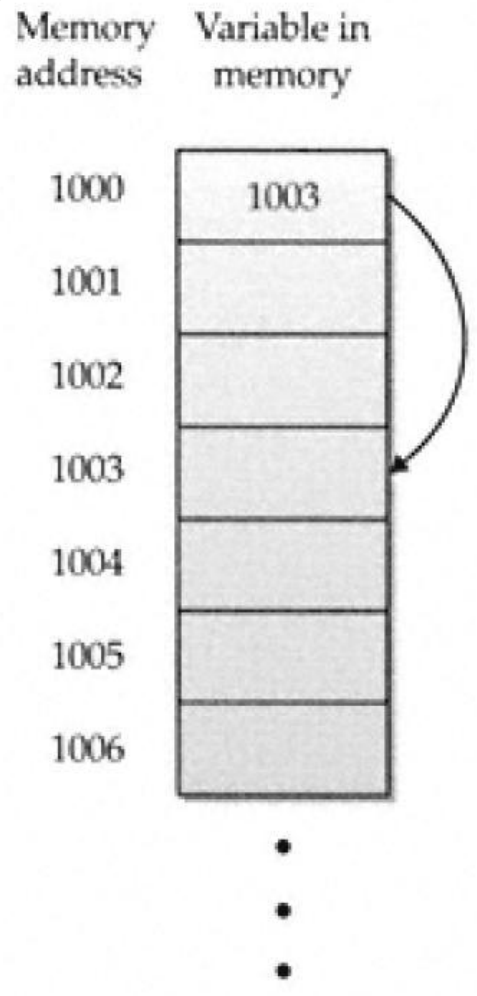
Fundamentals of Computer Programming

C Programming
4. Pointers

Contents

What Are Pointers?

- A *pointer* is a variable that holds a memory address.
- This address is the location of another object (typically another *variable*) in memory.
- If one variable contains the address of another variable, the first variable is said to *point to* the second.



Pointer Variables

- `type *name;`
- where *type* is the base type of the pointer, the name of the pointer variable is specified by *name*.
- All pointer operations are done relative to the pointer's base type.
- For example, when you declare a pointer to be of type **int ***, the compiler assumes that any address that it holds points to an integer.

Pointer Operators

- The **&** is a unary operator that returns *the memory address* of its operand.
  - `m = &count;`
- You can think of **&** as returning "**the address of.**“
- The second pointer operator, *****, returns *the value* located at the address that follows.
  - `q = *m;`
- You can think of ***** as “ **the value at address.**"

Pointer Expressions - Assignments

```
#include <stdio.h>

int main(void)
{
  int x = 99;
  int *p1, *p2;

  p1 = &x;
  p2 = p1;

  /* print the value of x twice */
  printf(''Values at p1 and p2: %d %
d\n", *p1, *p2);

  /* print the address of x twice */
  printf("Addresses pointed to by p1 and p2: %p %p", p1, p2);

  return 0;
}
```

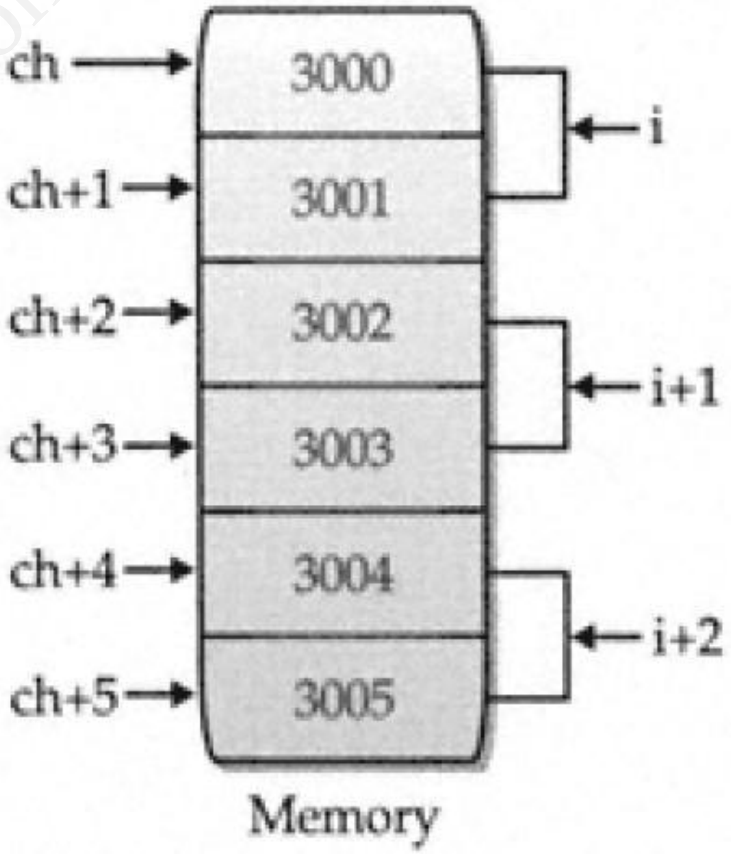
Pointer Expressions - Conversions

```
#include <stdio.h>

int main(void)
{
  double x = 100.1, y;
  int *p;

  /* The next statement causes p (which is an
      integer pointer) to point to a double. */
  p = (int *) &x;

  /* The next statement does not operate as expected. */
  y = *p; /* attempt to assign y the value x through p */

  /* The following statement won't output 100.1. */
  printf(''The (incorrect) value of x is: %f", y);

  return 0;
}
```

Pointer Expressions - Arithmetic

- Let **p** be an integer pointer with a current value of 2000.
- Assume **ints** are 4 bytes long.
- After the expression: **p++;** , **p** contains 2004.

```
char *ch = (char *) 3000;
int *i = (int *) 3000;
```

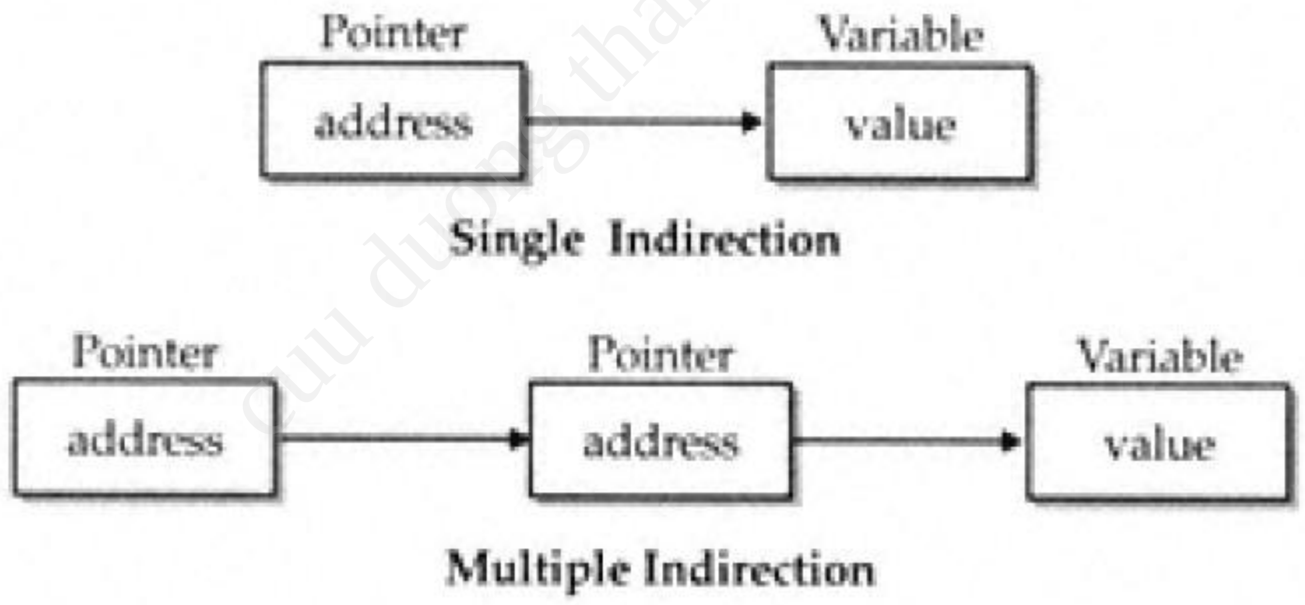


Pointer Expressions - Comparisons

- Ref: [2] pp. 126-127

Multiple Indirection – Pointers to Pointers

- In the case of a pointer to a pointer, the first pointer contains *the address of* the second pointer, which *points to* the object that contains the desired value.

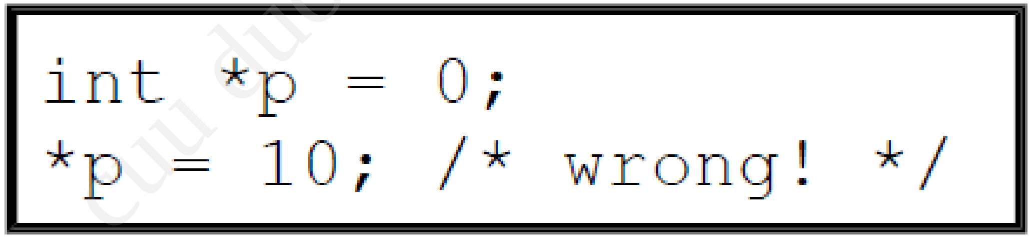


Multiple Indirection – Pointers to Pointers

```
#include <stdio.h>

int main(void)
{
  int x, *p, **q;

  x = 10;
  p = &x;
  q = &p;

  printf("%d", **q); /* print the value of x */

  return 0;
}
```

Initializing Pointers

- A pointer that does not currently point to a valid memory location is given the value null.
  - `char *p = 0;`
  - `char *p = NULL; // include <stdio.h>`



```
int *p = 0;
*p = 10; /* wrong! */
```