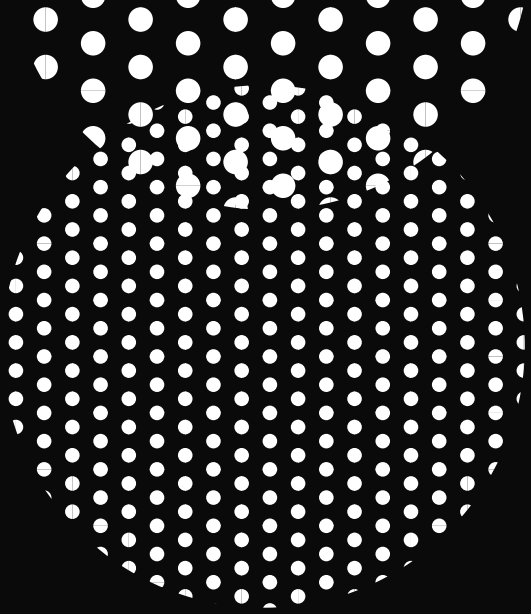

the new data type is Pointers

- int Pointers
- Char Pointers
- Float Pointers

declaring Pointers: `int *count;`

naming: (*Same name + Ptr)

reading Pointers → from right to left

linking the Variable to the Pointers:

`yPtr = &y;`

→ to get the address of the Variable

Printf("%d", y); → direct

Printf("%d", *yPtr); → Indirect

```c
// Fig. 7.4: fig07_04.c
// Using the & and * pointer operators.
#include <stdio.h>

int main(void)
{
   int a = 7;
   int *aPtr = &a; // set aPtr to the address of a

   printf("The address of a is %p"
          "\nThe value of aPtr is %p", &a, aPtr);

   printf("\n\nThe value of a is %d"
          "\nThe value of *aPtr is %d", a, *aPtr);

   printf("\n\nShowing that * and & are complements of "
          "each other\n&*aPtr = %p"
          "\n*&aPtr = %p\n", &*aPtr, *&aPtr);
}
```

address

value

address

(the address of) *& same &*

(the value of) cancel each other

Passing by reference → You can send back more than one number.

```c
// Cube a variable using pass-by-reference with a pointer argument.

#include <stdio.h>

void cubeByReference(int *nPtr); // function prototype

int main(void)
{
   int number = 5; // initialize number

   printf("The original value of number is %d", number);

   // pass address of number to cubeByReference
   cubeByReference(&number);

   printf("\nThe new value of number is %d\n", number);
}

// calculate cube of *nPtr; actually modifies number in main
void cubeByReference(int *nPtr)
{
   *nPtr = *nPtr * *nPtr * *nPtr; // cube *nPtr
}
```

int *nPtr = &number;

address → because → Passing by reference

No return because → Passing by reference

Fig. 7.7 | Cube a variable using pass-by-reference with a pointer argument. (Part 1 of 2.)

character at a time.

C standard library function touper from the <ctype.h> header is called to convert each character to its corresponding uppercase

while (*sPtr != '\0')

NULL Char