9-1 Introduction

A pointer is a constant or variable that contains an address that can be used to access data. Pointers are built on the basic concept of pointer constants.

<u>Topics discussed in this section:</u>

Pointer Constants
Pointer Values
Pointer Variables
Accessing Variables Through Pointers
Pointer Declaration and Definition
Declaration versus Redirection
Initialization of Pointer Variables

Note

An address expression, one of the expression types in the unary expression category, consists of an ampersand (&) and a variable name.

```c
// Print character addesses
#include <stdio.h>

int main (void)
{
// Local Declarations
   char a;
   char b;
// Statements
   printf ("%p\n %p\n", &a, &b);
   return 0;
}  // main
```

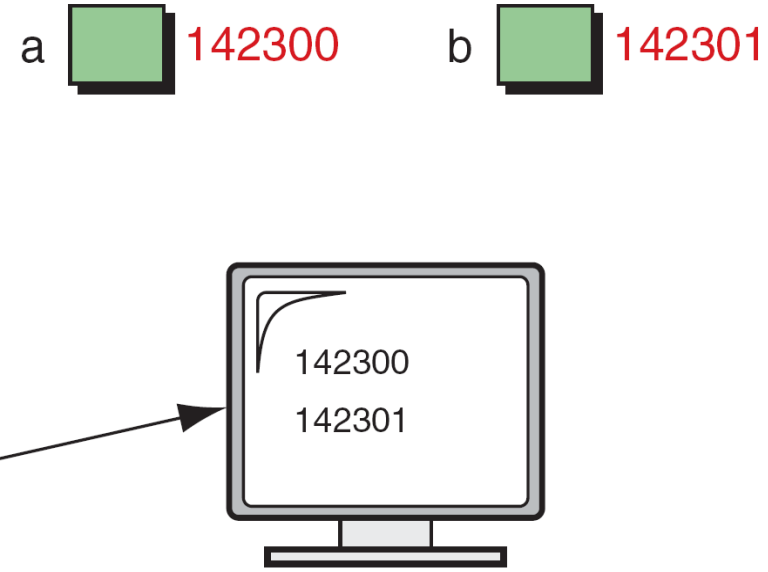


FIGURE 9-4 Print Character Addresses

Note

A variable's address is the first byte occupied by the variable.

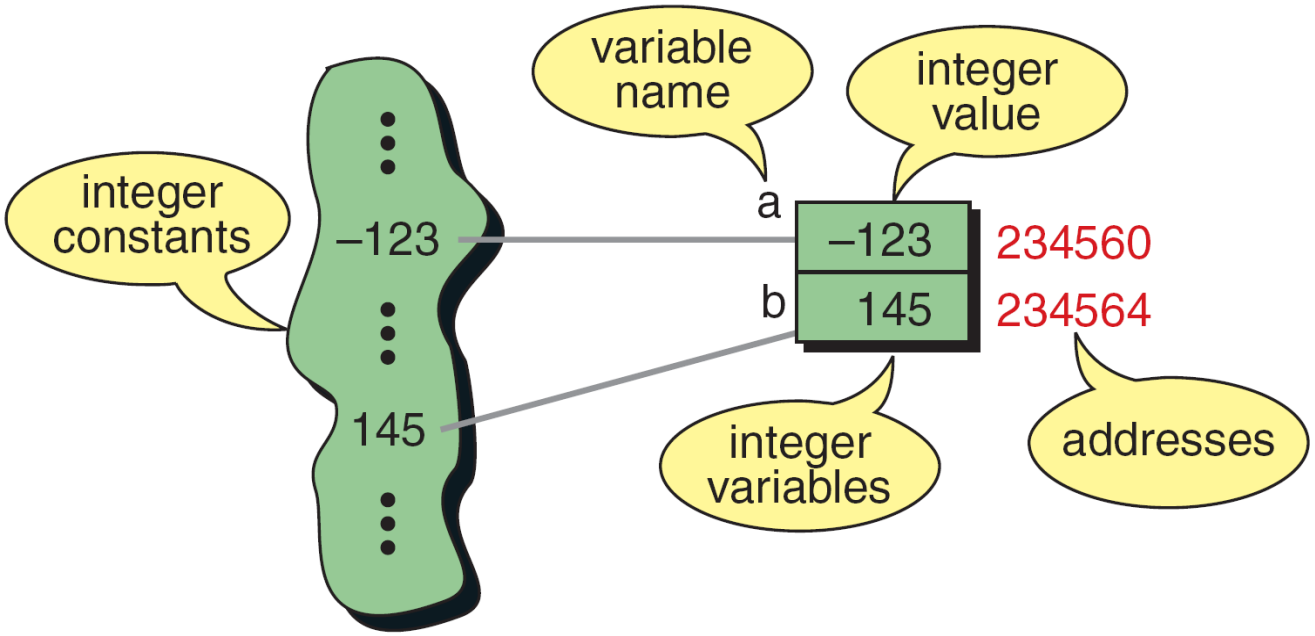


FIGURE 9-5 Integer Constants and Variables

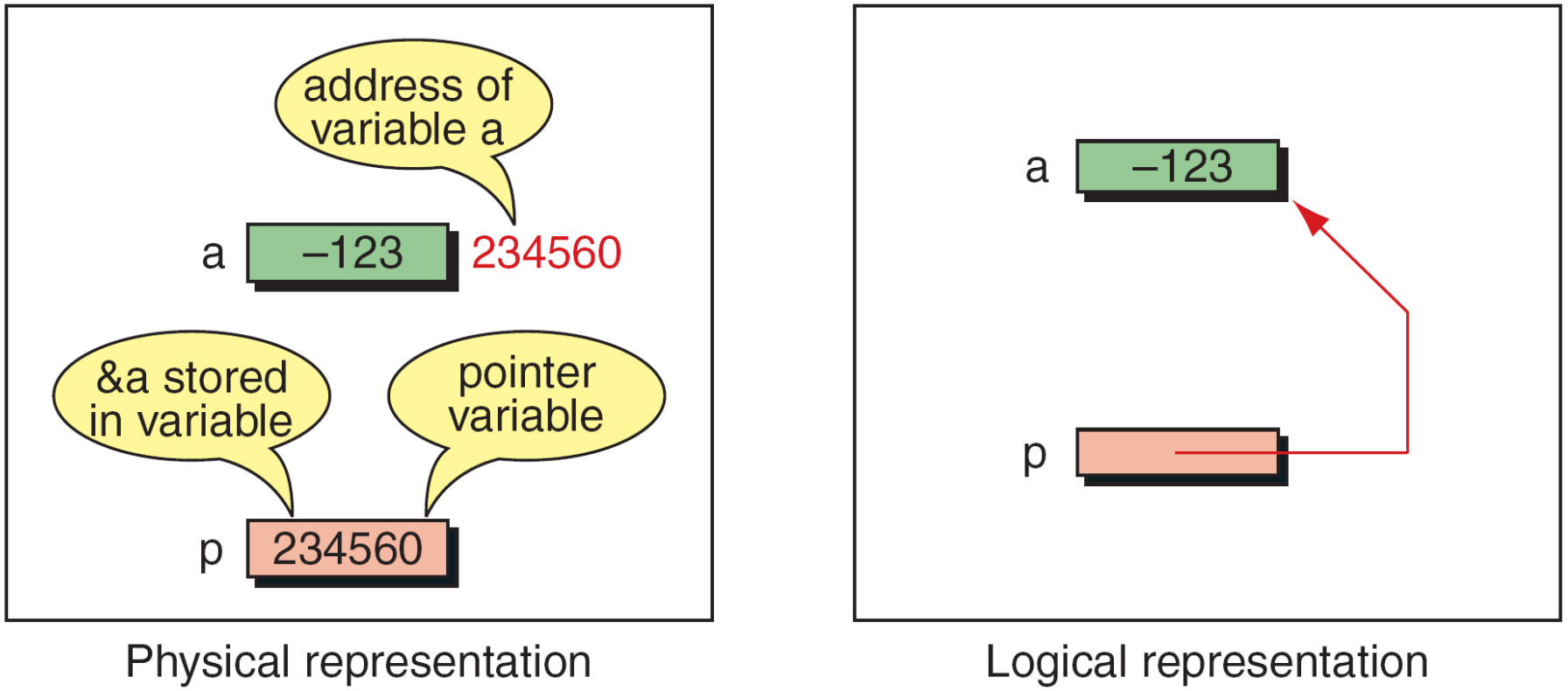


FIGURE 9-6 Pointer Variable

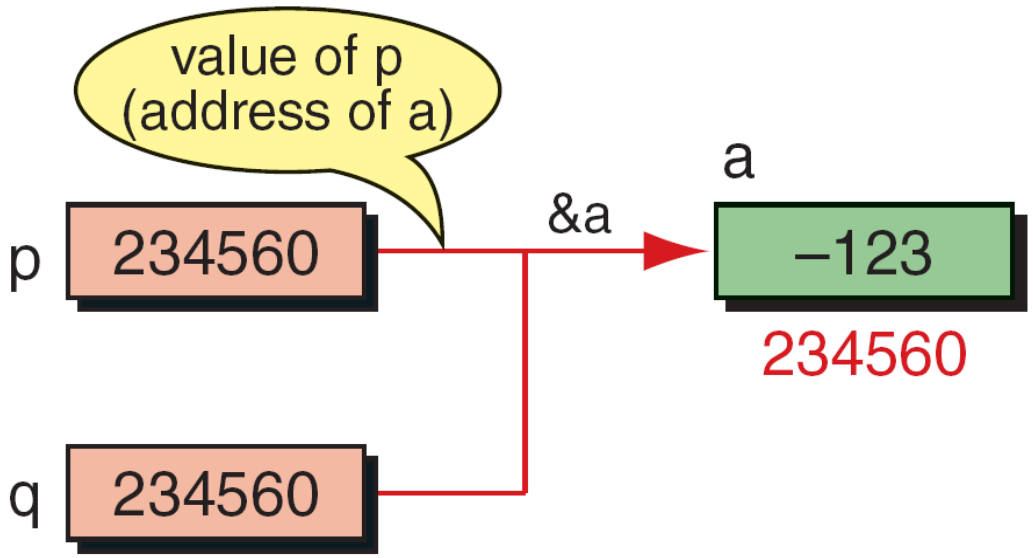


FIGURE 9-7 Multiple Pointers to a Variable

Note

A pointer that points to no variable contains the special null-pointer constant, NULL.

Note

An indirect expression, one of the expression types in the unary expression category, is coded with an asterisk (*) and an identifier.

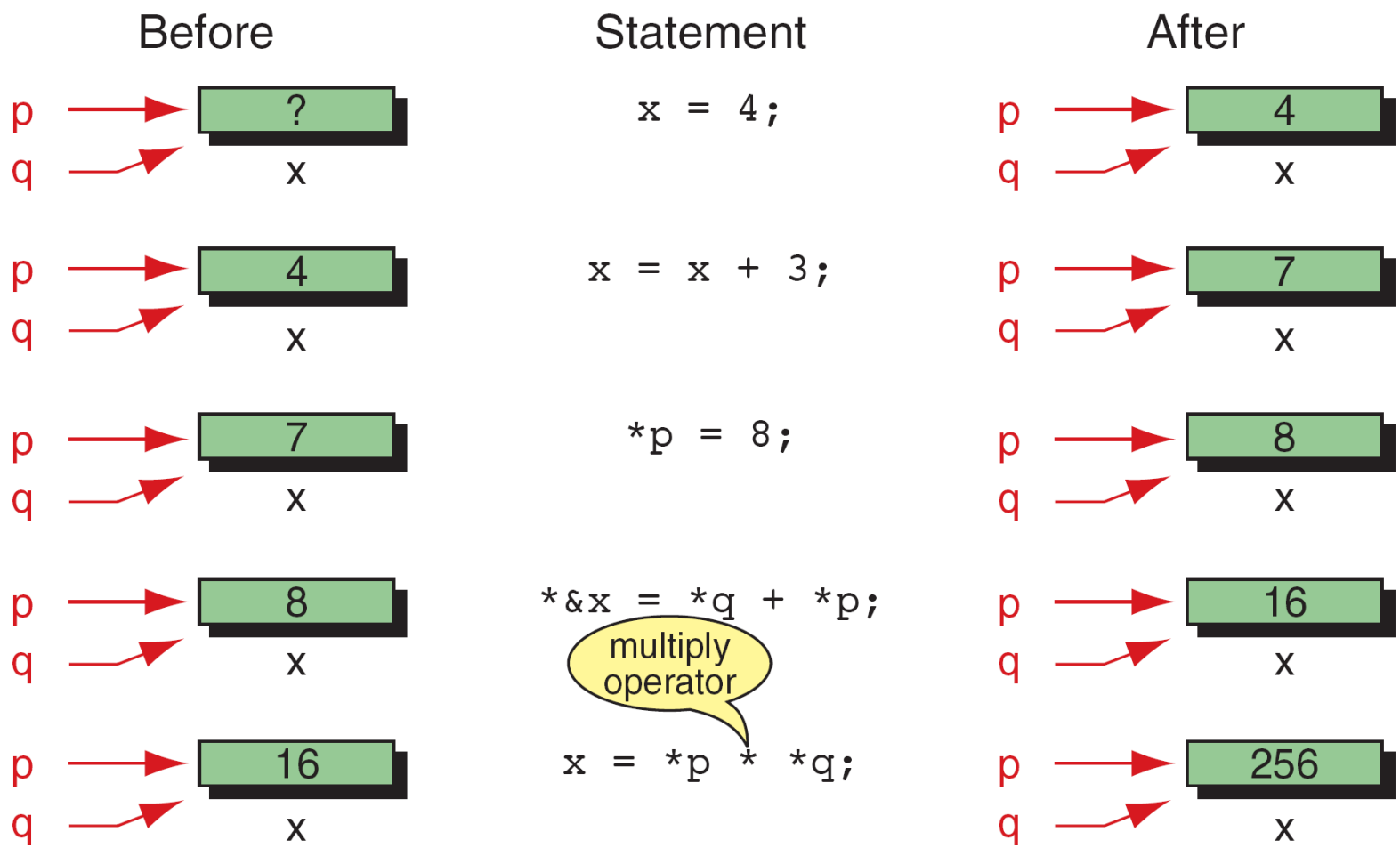


FIGURE 9-8 Accessing Variables Through Pointers

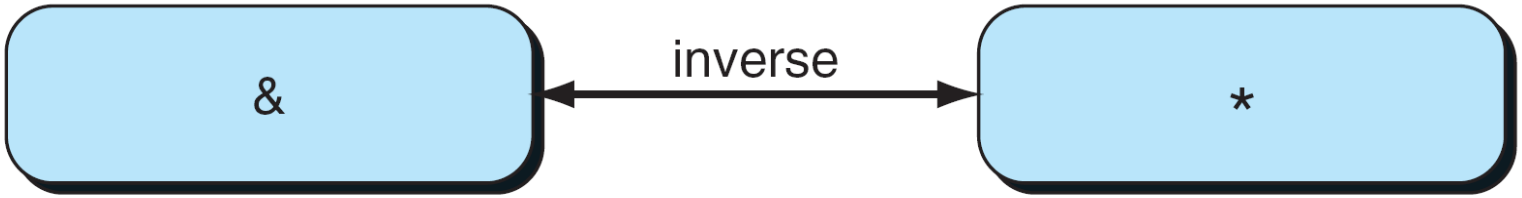


FIGURE 9-9 Address and Indirection Operators

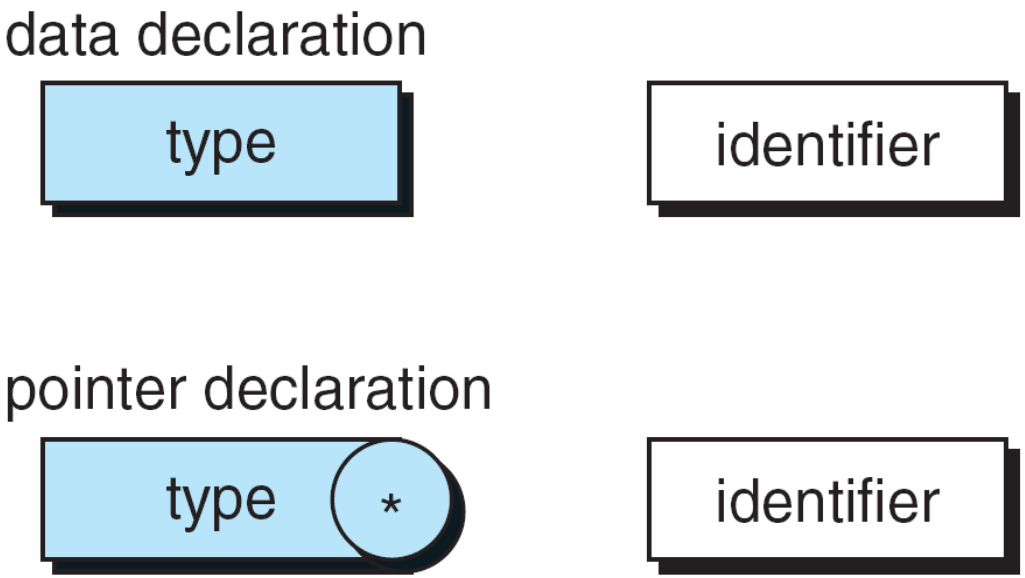


FIGURE 9-10 Pointer Variable Declaration

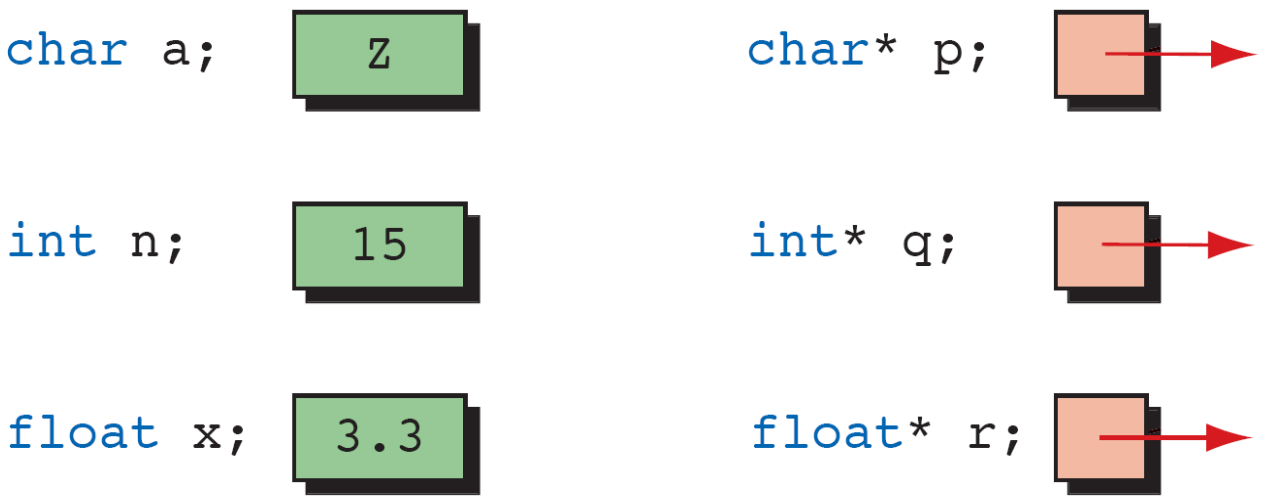
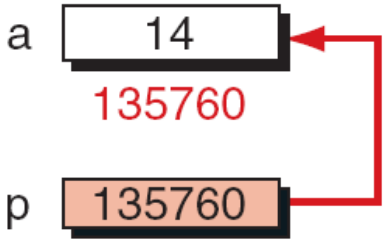


FIGURE 9-11 Declaring Pointer Variables

PROGRAM 9-1 Demonstrate Use of Pointers

```
 1  /* Demonstrate pointer use
 2        Written by:
 3        Date:
 4  */
 5  #include <stdio.h>
 6
 7  int main (void)
 8  {
 9  // Local Declarations
10     int  a;
11     int* p;
12
13  // Statements
14     a = 14;
15     p = &a;
16
17     printf("%d %p\n", a, &a);
```



PROGRAM 9-1 Demonstrate Use of Pointers

```
18      printf("%p %d %d\n",
19              p, *p, a);
20
21      return 0;
22  }  // main
```

```
Results:
14 00135760
00135760 14 14
```

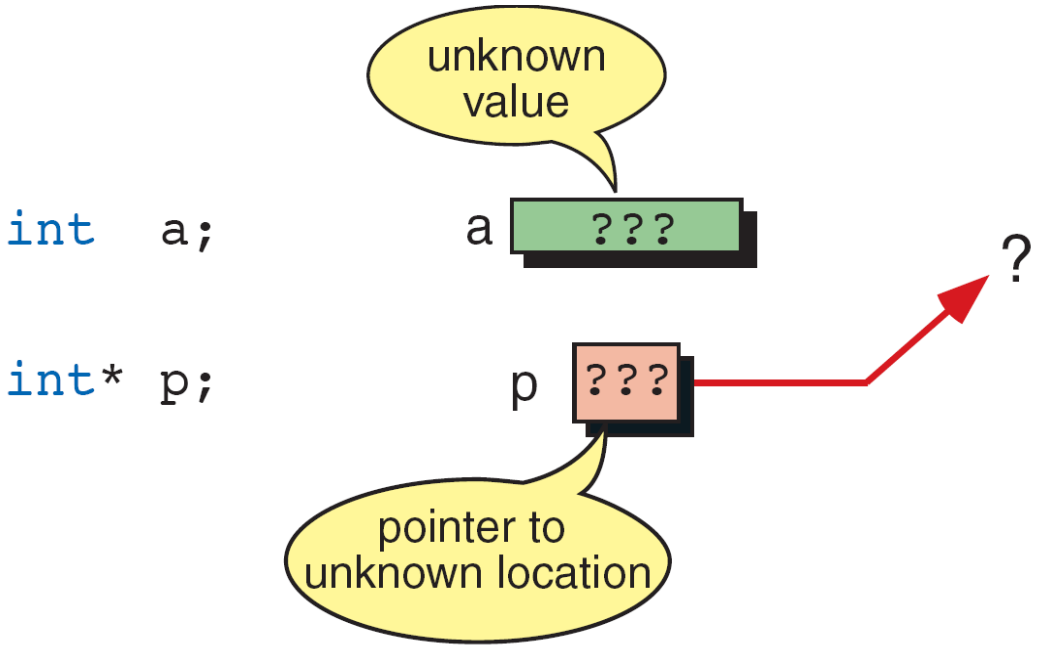


FIGURE 9-12 Uninitialized Pointers

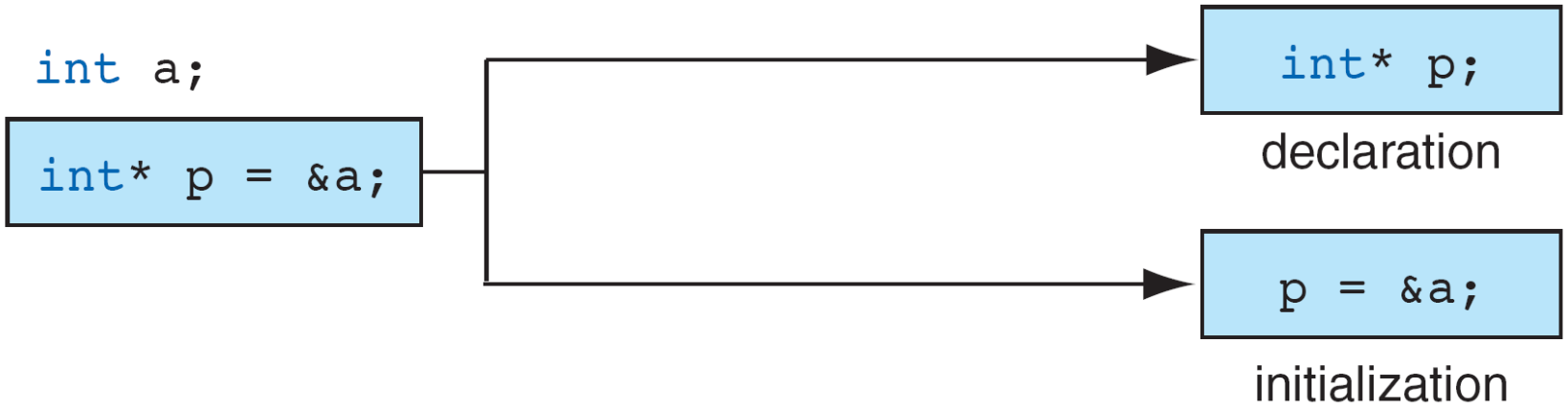
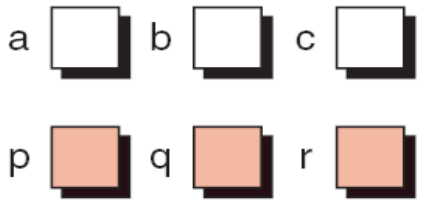
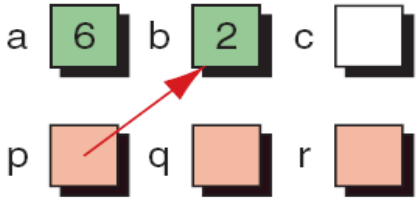


FIGURE 9-13 Initializing Pointer Variables

PROGRAM 9-2 Fun with Pointers

```
 1  /* Fun with pointers
 2        Written by:
 3        Date:
 4  */
 5  #include <stdio.h>
 6
 7  int main (void)
 8  {
 9  // Local Declarations
10     int  a;
11     int  b;
12     int  c;
13     int* p;
14     int* q;
15     int* r;
16
17  // Statements
18     a = 6;
19     b = 2;
20     p = &b;
21
```





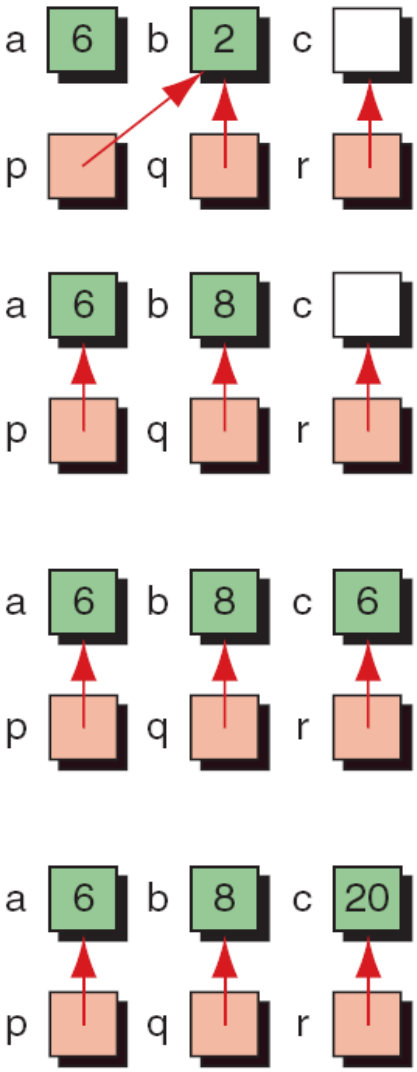
PROGRAM 9-2 Fun with Pointers

```
22      q = p;
23      r = &c;
24

25      p  = &a;
26      *q = 8;
27

28      *r = *p;
29

30      *r = a + *q + *&c;
31
32      printf("%d %d %d \n",
33                a, b, c);
```



PROGRAM 9-2 Fun with Pointers

```
34      printf("%d %d %d",
35              *p, *q, *r);
36      return 0;
37  }   // main
```

```
Results:
6 8 20
6 8 20
```

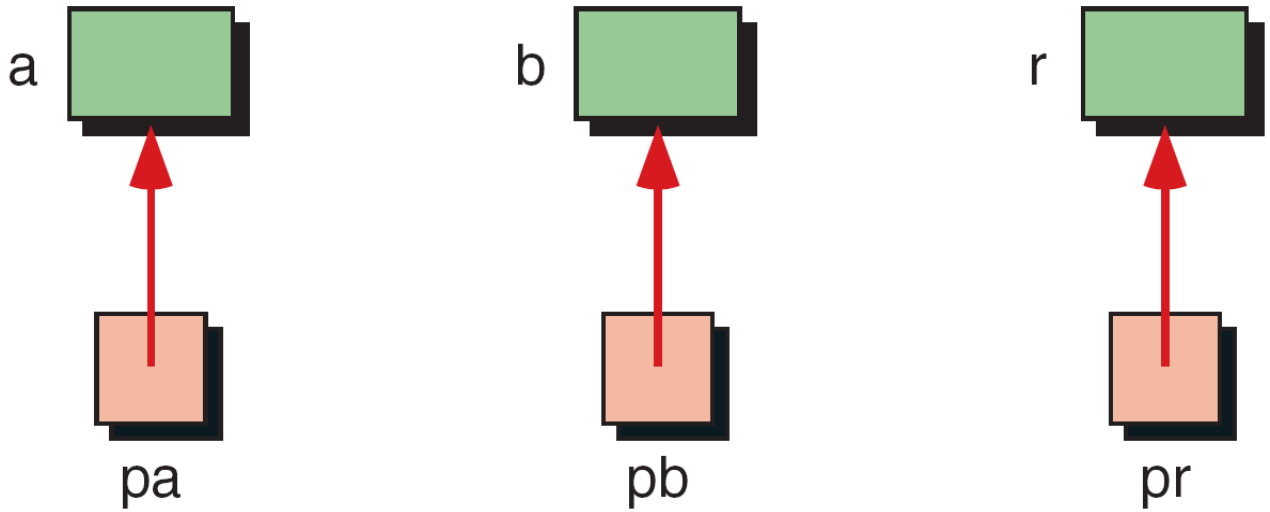


FIGURE 9-14 Add Two Numbers Using Pointers

PROGRAM 9-3 Add Two Numbers Using Pointers

```
1   /* This program adds two numbers using pointers to
2      demonstrate the concept of pointers.
3         Written by:
4         Date:
5   */
6   #include <stdio.h>
7
8   int main (void)
9   {
10  // Local Declarations
11     int  a;
12     int  b;
13     int  r;
14     int* pa = &a;
15     int* pb = &b;
16     int* pr = &r;
17
```

PROGRAM 9-3 Add Two Numbers Using Pointers

```
18  // Statements
19     printf("Enter the first number : ");
20     scanf ("%d", pa);
21     printf("Enter the second number: ");
22     scanf ("%d", pb);
23     *pr = *pa + *pb;
24     printf("\n%d + %d is %d", *pa, *pb, *pr);
25     return 0;
26  }  // main
```

```
Results:
   Enter the first number : 15
   Enter the second number: 51

   15 + 51 is 66
```

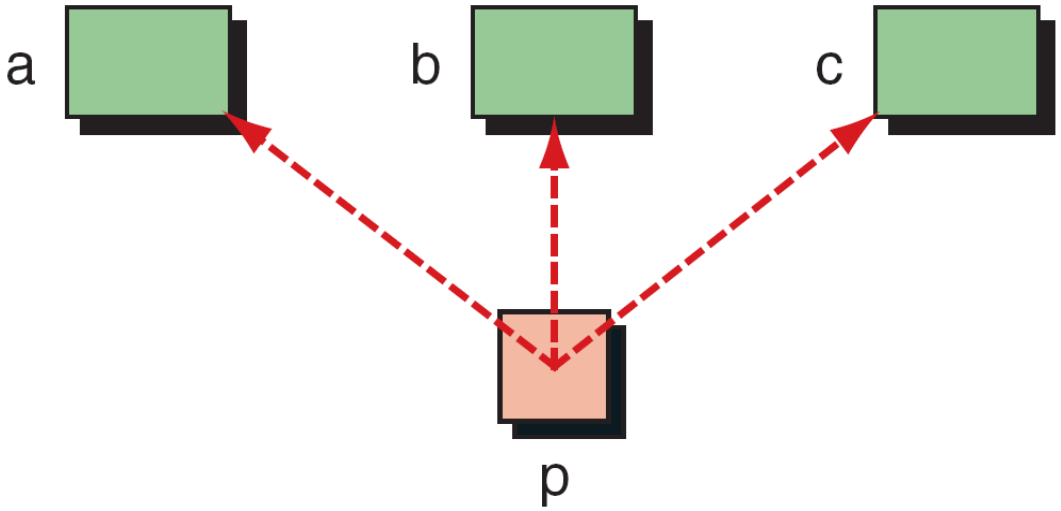


FIGURE 9-15 Demonstrate Pointer Flexibility

PROGRAM 9-4 Using One Pointer for Many Variables

```
1  /* This program shows how the same pointer can point to
2     different data variables in different statements.
3        Written by:
4        Date:
5  */
6  #include <stdio.h>
7
8  int main (void)
9  {
10 // Local Declarations
11    int  a;
12    int  b;
13    int  c;
14    int* p;
15
16 // Statements
17    printf("Enter three numbers and key return: ");
18    scanf ("%d %d %d", &a, &b, &c);
```

PROGRAM 9-4 Using One Pointer for Many Variables

```
19      p = &a;
20      printf("%3d\n", *p);
21      p = &b;
22      printf("%3d\n", *p);
23      p = &c;
24      printf("%3d\n", *p);
25      return 0;
26   }  // main
```

```
Results:
Enter three numbers and key return: 10 20 30
 10
 20
 30
```

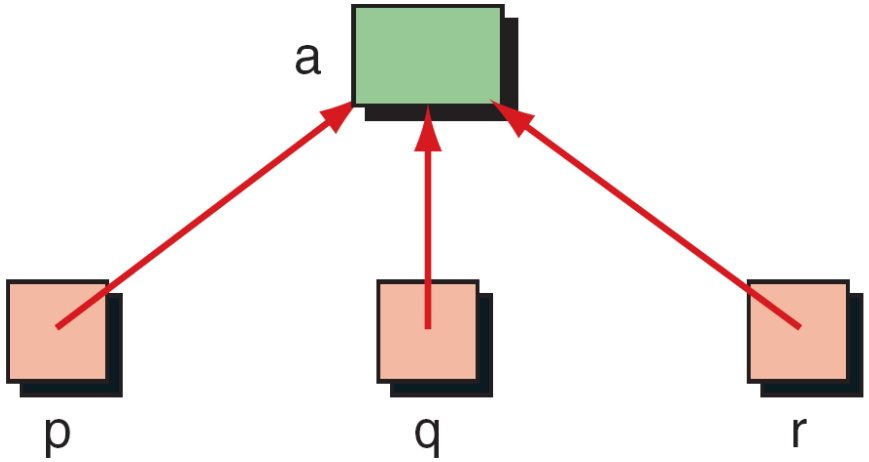


FIGURE 9-16 One Variable with Many Pointers

PROGRAM 9-5 Using A Variable with Many Pointers

```
1   /* This program shows how we can use different pointers
2      to point to the same data variable.
3         Written by:
4         Date:
5   */
6   #include <stdio.h>
7
8   int main (void)
9   {
10  // Local Declarations
11     int  a;
12     int* p = &a;
13     int* q = &a;
14     int* r = &a;
15
16  // Statements
17     printf("Enter a number: ");
18     scanf ("%d", &a);
```

PROGRAM 9-5 Using A Variable with Many Pointers

```
19      printf("%d\n", *p);
20      printf("%d\n", *q);
21      printf("%d\n", *r);
22
23      return 0;
24  }  // main
```

```
Results:
Enter a number: 15
15
15
15
```