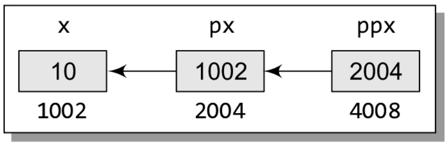

Figure 1.12 Pointer to pointer

For example, consider the following code:

```
int x=10;
int *px, **ppx;
px = &x;
ppx = &px;
```

Let us assume, the memory locations of these variables are as shown in Fig. 1.12.

Now if we write,

```
printf("\n %d", **ppx);
```

Then, it would print 10, the value of `x`.

1.11.6 Drawbacks of Pointers

Although pointers are very useful in C, they are not free from limitations. If used incorrectly, pointers can lead to bugs that are difficult to unearth. For example, if you use a pointer to read a memory location but that pointer is pointing to an incorrect location, then you may end up reading a wrong value. An erroneous input always leads to an erroneous output. Thus however efficient your program code may be, the output will always be disastrous. Same is the case when writing a value to a particular memory location.

Let us try to find some common errors when using pointers.

```
int x, *px;
x=10;
*px = 20;
```

Error: Un-initialized pointer. `px` is pointing to an unknown memory location. Hence it will overwrite that location's contents and store 20 in it.

```
int x, *px;
x=10;
px = x;
```

Error: It should be `px = &x;`

```
int x=10, y=20, *px, *py;
px = &x, py = &y;
if(px<py)
printf("\n x is less than y");
else
printf("\n y is less than x");
```

Error: It should be `if(*px< *py)`

Points to Remember

- C was developed in the early 1970s by Dennis Ritchie at Bell Laboratories.
- Every word in a C program is either an identifier or a keyword. Identifiers are the names given to program elements such as variables and functions. Keywords are reserved words which cannot be used as identifiers.
- C provides four basic data types: `char`, `int`, `float`, and `double`.
- A variable is defined as a meaningful name given to a data storage location in computer memory.
- Standard library function `scanf()` is used to input data in a specified format.`printf()`function is used to output data of different types in a specified format.
- C supports different types of operators which can be classified into following categories: arithmetic, relational, equality, logical, unary, conditional, bitwise, assignment, comma, and sizeof operators.

- Modulus operator (%) can only be applied on integer operands, and not on float or double operands.
- Equality operators have lower precedence than relational operators.
- Like arithmetic expressions, logical expressions are evaluated from left to right.
- Both x++ and ++x increment the value of x, but in the former case, the value of x is returned before it is incremented. Whereas in the latter case, the value of x is returned after it is incremented.
- Conditional operator is also known as ternary operator as it takes three operands.
- Bitwise NOT or complement produces one's complement of a given binary number.
- Among all the operators, comma operator has the lowest precedence.
- `sizeof` is a unary operator used to calculate the size of data types. This operator can be applied to all data types.
- While type conversion is done implicitly, typecasting has to be done explicitly by the programmer. Typecasting is done when the value of one data type has to be converted into the value of another data type.
- C supports three types of control statements: decision control statements, iterative statements, and jump statements.
- In a `switch` statement, if the value of the variable does not match with any of the values of case statements, then default case is executed.
- Iterative statements are used to repeat the execution of a list of statements until the specified expression becomes false.
- The `break` statement is used to terminate the execution of the nearest enclosing loop in which it appears.
- When the compiler encounters a `continue` statement, then the rest of the statements in the loop are skipped and the control is unconditionally transferred to the loop-continuation portion of the nearest enclosing loop.
- A C program contains one or more functions, where each function is defined as a group of statements that perform a specific task.
- Every C program contains a `main()` function which is the starting point of the program. It is the function that is called by the operating system when the user runs the program.
- Function declaration statement identifies a function's name and the list of arguments that it accepts and the type of data it returns.
- Function definition, on the other hand, consists of a function header that identifies the function, followed by the body of the function containing the executable code for that function. When a function is defined, space is allocated for that function in the memory.
- The moment the compiler encounters a function call, the control jumps to the statements that are a part of the called function. After the called function is executed, the control is returned back to the calling function.
- Placing the function declaration statement prior to its use enables the compiler to make a check on the arguments used while calling that function.
- A function having `void` as its return type cannot return any value. Similarly, a function having `void` as its parameter list cannot accept any value.
- Call by value method passes values of the variables to the called function. Therefore, the called function uses a copy of the actual arguments to perform its intended task. This method is used when the function does not need to modify the values of the original variables in the calling function.
- In call by reference method, addresses of the variables are passed by the calling function to the called function. Hence, in this method, a function receives an implicit reference to the argument, rather than a copy of its value. This allows the function to modify the value of the variable and that change is reflected in the calling function as well.
- A pointer is a variable that contains the memory address of another variable.
- The & operator retrieves the address of the variable.
- We can 'dereference' a pointer, i.e., refer to the value of the variable to which it points by using unary * operator.
- Null pointer is a special pointer variable that does not point to any variable. This means that a null pointer does not point to any valid memory address. To declare a null pointer we may use the predefined constant NULL.
- A generic pointer is pointer variable that has `void` as its data type. The generic pointer can point to variables of any data type.
- To declare pointer to pointers, we need to add an asterisk (*) for each level of reference.

Exercises

Review Questions

1. Discuss the structure of a C program.
2. Differentiate between declaration and definition.
3. How is memory reserved using a declaration statement?
4. What do you understand by identifiers and keywords?
5. Explain the terms variables and constants. How many types of variables are supported by C?
6. What does the data type of a variable signify?
7. Write a short note on basic data types that the C language supports.
8. Why do we include <stdio.h> in our programs?
9. What are header files? Explain their significance.
10. Write short notes on printf and scanf functions.
11. Write a short note on operators available in C language.
12. Draw the operator precedence chart.
13. Differentiate between typecasting and type conversion.
14. What are decision control statements? Explain in detail.
15. Write a short note on the iterative statements that C language supports.
16. When will you prefer to work with a switch statement?
17. Define function. Why are they needed?
18. Differentiate between function declaration and function definition.
19. Why is function declaration statement placed prior to function definition?
20. Explain the concept of making function calls.
21. Differentiate between call by value and call by reference using suitable examples.
22. Write a short note on pointers.
23. Explain the difference between a null pointer and a void pointer.
24. How are generic pointers different from other pointer variables?
25. Write a short note on pointers to pointers.

Programming Exercises

1. Write a program to read 10 integers. Display these numbers by printing three numbers in a line separated by commas.
2. Write a program to print the count of even numbers between 1–200. Also print their sum.
3. Write a program to count the number of vowels in a text.
4. Write a program to read the address of a user. Display the result by breaking it in multiple lines.
5. Write a program to read two floating point numbers. Add these numbers and assign the result to an integer. Finally, display the value of all the three variables.
6. Write a program to read a floating point number. Display the rightmost digit of the integral part of the number.
7. Write a program to calculate simple interest and compound interest.
8. Write a program to calculate salary of an employee given his basic pay (to be entered by the user), HRA = 10% of the basic pay, TA = 5% of basic pay. Define HRA and TA as constants and use them to calculate the salary of the employee.
9. Write a program to prepare a grocery bill. Enter the name of the items purchased, quantity in which it is purchased, and its price per unit. Then display the bill in the following format:

```
************ B I L L ************
   Item  Quantity  Price  Amount
 _________________________________

 _________________________________
      Total Amount to be paid
 _________________________________
```

10. Write a C program using printf statement to print BYE in the following format:

```
BBB      Y     Y       EEEE
B   B      Y   Y       E
BBB          Y         EEEE
B   B        Y
```

11. Write a program to read an integer. Display the value of that integer in decimal, octal, and hexadecimal notation.
12. Write a program that prints a floating point value in exponential format with the following specifications:
    (a) correct to two decimal places;
    (b) correct to four decimal places; and

(c) correct to eight decimal places.

13. Write a program to find the smallest of three integers using functions.
14. Write a program to calculate area of a triangle using function.
15. Write a program to find whether a number is divisible by two or not using functions.
16. Write a program to print 'Programming in C is Fun' using pointers.
17. Write a program to read a character and print it. Also print its ASCII value. If the character is in lower case, print it in upper case and vice versa. Repeat the process until a '*' is entered.
18. Write a program to add three floating point numbers. The result should contain only two digits after the decimal.
19. Write a program to take input from the user and then check whether it is a number or a character. If it is a character, determine whether it is in upper case or lower case. Also print its ASCII value.
20. Write a program to display sum and average of numbers from `1` to `n`. Use `for` loop.
21. Write a program to print all odd numbers from `m` to `n`.
22. Write a program to print all prime numbers from `m` to `n`.
23. Write a program to read numbers until –1 is entered and display whether it is an Armstrong number or not.
24. Write a program to add two floating point numbers using pointers and functions.
25. Write a program to calculate area of a triangle using pointers.

Multiple-choice Questions

1. The operator which compares two values is
   (a) Assignment (b) Relational
   (c) Unary (d) Equality
2. Ternary operator operates on how many operands?
   (a) 1 (b) 2
   (c) 3 (d) 4
3. Which operator produces the one's complement of the given binary value?
   (a) Logical AND (b) Bitwise AND
   (c) Logical OR (d) Bitwise NOT
4. Which operator has the lowest precedence?
   (a) Sizeof (b) Unary
   (c) Assignment (d) Comma
5. Which of the following is the conversion character associated with short integer?
   (a) `%c` (b) `%h`
   (c) `%e` (d) `%f`
6. Which of the following is not a character constant?
   (a) `'A'` (b) "A"
   (c) `' '` (d) `'*'`
7. Which of the following is a valid variable name?
   (a) Initial.Name (b) A+B
   (c) $amt (d) Floats
8. Which operator cannot be used with floating point numbers?
   (a) `+` (b) `-`
   (c) `%` (d) `*`
9. Identify the erroneous expression.
   (a) `X=y=2, 4;` (b) `res = ++a * 5;`
   (c) `res = /4;` (d) `res = a++ -b *2`
10. Function declaration statement identifies a function with its
    (a) Name
    (b) Arguments
    (c) Data type of return value
    (d) All of these
11. Which return type cannot return any value to the calling function?
    (a) int (b) float
    (c) void (d) double
12. Memory is allocated for a function when the function is
    (a) declared (b) defined
    (c) called (d) returned
13. `*(&num)` is equivalent to writing
    (a) `&num` (b) `*num`
    (c) `num` (d) None of these
14. Which operator retrieves the `lvalue` of a variable?
    (a) `&` (b) `*`
    (c) `->` (d) None of these
15. Which operator is used to dereference a pointer?
    (a) `&` (b) `*`
    (c) `->` (d) None of these

True or False

1. We can have only one function in a C program.
2. Keywords are case sensitive.
3. Variable 'first' is the same as 'First'.
4. Signed variables can increase the maximum positive range.
5. `Comment` statements are not executed by the compiler.

6. Equality operators have higher precedence than the relational operators.
7. Shifting once to the left multiplies the number by 2.
8. `Decision control` statements are used to repeat the execution of a list of statements.
9. `printf("%d", scanf("%d", &num));` is a valid C statement.
10. 1,234 is a valid integer constant.
11. A `printf` statement can generate only one line of output.
12. `stdio.h` is used to store the source code of the program.
13. The closing brace of `main()` is the logical end of the program.
14. The declaration section gives instructions to the computer.
15. Any valid printable ASCII character can be used for a variable name.
16. Underscore can be used anywhere in the variable name.
17. `void` is a data type in C.
18. All arithmetic operators have same precedence.
19. The modulus operator can be used only with integers.
20. The calling function always passes parameters to the called function.
21. The name of a function is global.
22. No function can be declared within the body of another function.
23. The `&` operator retrieves the `lvalue` of the variable.
24. Unary increment and decrement operators have greater precedence than the dereference operator.
25. On 32-bit systems, an integer variable is allocated 4 bytes.

Fill in the Blanks

1. C was developed by ______.
2. The execution of a C program begins at ______.
3. In the memory, characters are stored as ______.
4. `return 0` returns 0 to the ______.
5. ______ finds the remainder of an integer division.
6. `sizeof` is a ______ operator used to calculate the sizes of data types.
7. ______ is also known as forced conversion.
8. ______ is executed when the value of the variable does not match with any of the values of the case statement.
9. ______ function prints data on the monitor.
10. A C program ends with a ______.
11. ______ causes the cursor to move to the next line.
12. A variable can be made constant by declaring it with the qualifier ______ at the time of initialization.
13. ______ operator returns the number of bytes occupied by the operand.
14. The ______ specification is used to read/write a short integer.
15. The ______ specification is used to read/write a hexadecimal integer.
16. To print the data left-justified, ______ specification is used.
17. After the function is executed, the control passes back to the ______.
18. A function that uses another function is known as the ______.
19. The inputs that the function takes are known as ______.
20. Function definition consist of ______ and ______.
21. In ______ method, address of the variable is passed by the calling function to the called function.
22. Size of character pointer is ______.
23. ______ pointer does not point to any valid memory address.
24. The ______ appears on the right side of the assignment statement.
25. The ______ operator informs the compiler that the variable is a pointer variable.