22 Bits, Characters, C Strings and structs

Objectives

In this chapter you'll learn:

- To create and use `struct`s and to understand their near equivalence with classes.
- To use `typedef` to create aliases for data types.
- To manipulate data with the bitwise operators and to create bit fields for storing data compactly.
- To use the functions of the character-handling library `<cctype>`.
- To use the string-conversion functions of the general-utilities library `<cstdlib>`.
- To use the string-processing functions of the string-handling library `<cstring>`.

Self-Review Exercises

22.1 Fill in the blanks in each of the following:

a) The bits in the result of an expression using the ________ operator are set to one if the corresponding bits in each operand are set to one. Otherwise, the bits are set to zero.

ANS: bitwise AND (`&`).

b) The bits in the result of an expression using the ________ operator are set to one if at least one of the corresponding bits in either operand is set to one. Otherwise, the bits are set to zero.

ANS: bitwise inclusive OR (`|`).

c) Keyword ________ introduces a structure declaration.

ANS: `struct`.

d) Keyword ________ is used to create a synonym for a previously defined data type.

ANS: `typedef`.

e) Each bit in the result of an expression using the ________ operator is set to one if exactly one of the corresponding bits in either operand is set to one.

ANS: bitwise exclusive OR (`^`).

f) The bitwise AND operator `&` is often used to ________ bits (i.e., to select certain bits from a bit string while zeroing others).

ANS: mask.

g) The ________ and ________ operators are used to shift the bits of a value to the left or to the right, respectively.

ANS: left-shift operator (`<<`), right-shift operator (`>>`).

22.2 Write a single statement or a set of statements to accomplish each of the following:

a) Define a structure called `Part` containing `int` variable `partNumber` and `char` array `partName`, whose values may be as long as 25 characters.

ANS:
```
struct Part {
   int partNumber;
   char partName[26];
};
```

b) Define `PartPtr` to be a synonym for the type `Part*`.

ANS: `typedef Part* PartPtr;`

c) Use separate statements to declare variable `a` to be of type `Part`, array `b[10]` to be of type `Part` and variable `ptr` to be of type pointer to `Part`.

ANS:
```
Part a;
Part b[10];
Part* ptr;
```

d) Read a part number and a part name from the keyboard into the members of variable `a`.

ANS: `cin >> a.partNumber >> a.partName;`

e) Assign the member values of variable `a` to element three of array `b`.

ANS: `b[3] = a;`

f) Assign the address of array `b` to the pointer variable `ptr`.

ANS: `ptr = b;`

g) Print the member values of element three of array `b`, using the variable `ptr` and the structure pointer operator to refer to the members.

ANS:
```
cout << (ptr + 3)->partNumber << ' '
     << (ptr + 3)->partName << endl;
```

22.3 Write a single statement to accomplish each of the following. Assume that variables `c` (which stores a character), `x`, `y` and `z` are of type `int`; variables `d`, `e` and `f` are of type `double`; variable `ptr` is of type `char*` and arrays `s1[100]` and `s2[100]` are of type `char`.

a) Convert the character stored in `c` to an uppercase letter. Assign the result to variable `c`.

ANS: `c = toupper(c);`

b) Determine if the value of variable `c` is a digit. Use the conditional operator as shown in Figs. 22.18–22.20 to print `" is a "` or `" is not a "` when the result is displayed.

ANS:
```
cout << '\'' << c << "\' "
          << (isdigit(c) ? "is a" : "is not a")
           << " digit" << endl;
```

c) Determine whether the value of variable `c` is a control character. Use the conditional operator to print `" is a "` or `" is not a "` when the result is displayed.

ANS:
```
cout << '\'' << c << "\' "
          << (iscntrl(c) ? "is a"  : "is not a")
           << " control character" << endl;
```

d) Assign to `ptr` the location of the last occurrence of `c` in `s1`.

ANS: `ptr = strrchr(s1, c);`

e) Convert the string `"8.63582"` to `double`, and print the value.

ANS: `out << atof("8.63582") << endl;`

f) Determine whether the value of `c` is a letter. Use the conditional operator to print `" is a "` or `" is not a "` when the result is displayed.

ANS:
```
cout << '\'' << c << "\' "
           << (isalpha(c) ? "is a" : "is not a")
           << " letter" << endl;
```

g) Assign to `ptr` the location of the first occurrence of `s2` in `s1`.

ANS: `ptr = strstr(s1, s2);`

h) Determine whether the value of variable `c` is a printing character. Use the conditional operator to print `" is a "` or `" is not a "` when the result is displayed.

ANS:
```
cout << '\'' << c << "\' "
           << (isprint(c) ? "is a" : "is not a")
           << " printing character" << endl;
```

i) Assign to `ptr` the location of the first occurrence in `s1` of any character from `s2`.

ANS: `ptr = strpbrk(s1, s2);`

j) Assign to `ptr` the location of the first occurrence of `c` in `s1`.

ANS: `ptr = strchr(s1, c);`

k) Convert the string `"-21"` to `int`, and print the value.

ANS: `cout << atoi("-21") << endl;`

Exercises

NOTE: Solutions to the programming exercises are located in the **`ch22solutions`** *folder.*

22.4 *(Defining Structures)* Provide the definition for each of the following structures:

a) Structure `Inventory`, containing character array `partName[30]`, integer `partNumber`, floating-point `price`, integer `stock` and integer `reorder`.

ANS:
```
struct Inventory
{
    char partName[30];
    int partNumber;
    double price;
    int stock;
    int reorder;
};
```

b) A structure called `Address` that contains character arrays `streetAddress[25]`, `city[20]`, `state[3]` and `zipCode[6]`.

ANS:
```
struct Address
{
    char streetAddress[25];
    char city[20];
    char state[3];
    char zipCode[6];
};
```

c) Structure `Student`, containing arrays `firstName[15]` and `lastName[15]` and variable `homeAddress` of type `struct Address` from part (b).

ANS:
```
struct Student
{
    char firstName[15];
    char lastName[15];
    struct Address homeAddress;
};
```

d) Structure `Test`, containing 16 bit fields with widths of 1 bit. The names of the bit fields are the letters `a` to `p`.

ANS:
```
struct Test
{
    unsigned a:1, b:1, c:1, d:1, e:1, f:1, g:1, h:1,
       i:1, j:1, k:1, l:1, m:1, n:1, o:1, p:1;
};
```