APPENDIX K

Unions

A union, in almost all regards, is just like a structure. The difference is that all the members of a union use the same memory area, so only one member can be used at a time. A union might be used in an application where the program needs to work with two or more values (of different data types), but only needs to use one of the values at a time. Unions conserve memory by storing all their members in the same memory location.

Unions are declared just like structures, except the key word `union` is used instead of `struct`. Here is an example:

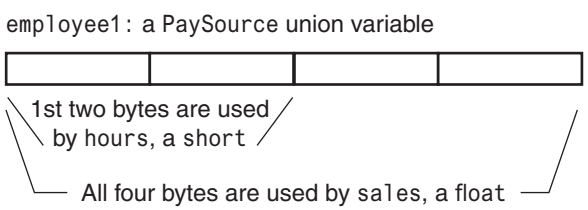

```
union PaySource
{
   short hours;
   float sales;
};
```

A union variable of the data type shown above can then be defined as

```
PaySource employee1;
```

The `PaySource` union variable defined here has two members: `hours` (a `short`), and `sales` (a `float`). The entire variable will only take up as much memory as the largest member (in this case, a `float`). The way this variable is stored on a typical PC is illustrated in Figure K-1.

Figure K-1



As shown in Figure K-1, the union uses four bytes on a typical PC. It can store a `short` or a `float`, depending on which member is used. When a value is stored in the `sales` member, all four bytes are needed to hold the data. When a value is stored in the `hours` member, however, only the first two bytes are used. Obviously, both members can't hold values at the same time. This union is demonstrated in Program K-1.

Program K-1

```
// This program demonstrates a union.
#include <iostream>
#include <iomanip>
using namespace std;

union PaySource
{
    int hours;        // Hours worked
    float sales;      // Amount of sales
};

int main()
{
    PaySource employee1;    // Define a union variable
    char payType;           // To hold the pay type
    float payRate;          // Hourly pay rate
    float grossPay;         // Gross pay

    cout << fixed << showpoint << setprecision(2);
    cout << "This program calculates either hourly wages or\n";
    cout << "sales commission.\n";

    // Get the pay type, hourly or commission.
    cout << "Enter H for hourly wages or C for commission: ";
    cin >> payType;

    // Determine the gross pay, depending on the pay type.
    if (payType == 'H' || payType == 'h')
    {
        // This is an hourly paid employee. Get the
        // pay rate and hours worked.
        cout << "What is the hourly pay rate? ";
        cin >> payRate;
        cout << "How many hours were worked? ";
        cin >> employee1.hours;

        // Calculate and display the gross pay.
        grossPay = employee1.hours * payRate;
        cout << "Gross pay: $" << grossPay << endl;
    }
    else if (payType == 'C' || payType == 'c')
    {
        // This is a commission-paid employee. Get the
        // amount of sales.
        cout << "What are the total sales for this employee? ";
        cin >> employee1.sales;

        // Calculate and display the gross pay.
        grossPay = employee1.sales * 0.10;
        cout << "Gross pay: $" << grossPay << endl;
    }
```

```
52      else
53      {
54          // The user made an invalid selection.
55          cout << payType << " is not a valid selection.\n";
56      }
57      return 0;
58  }
```

Program Output with Example Input Shown in Bold

```
This program calculates either hourly wages or
sales commission.
Enter H for hourly wages or C for commission: C Enter
What are the total sales for this employee? 5000 Enter
Gross pay: $500.00
```

Program Output with Different Example Input Shown in Bold

```
This program calculates either hourly wages or
sales commission.
Enter H for hourly wages or C for commission: H Enter
What is the hourly pay rate? 20 Enter
How many hours were worked? 40 Enter
Gross pay: $800.00
```


Everything else you already know about structures applies to unions. For example, arrays of unions may be defined. A union may be passed as an argument to a function or returned from a function. Pointers to unions may be defined, and the members of the union referenced by the pointer can be accessed with the -> operator.

Anonymous Unions

The members of an anonymous union have names, but the union itself has no name. Here is the general format of an anonymous union declaration:

```
union
{
     member declaration;
     ...
};
```

An anonymous union declaration actually creates the member variables in memory, so there is no need to separately define a union variable. Anonymous unions are simple to use because the members may be accessed without the dot operator. Program K-2, which is a modification of Program K-1, demonstrates the use of an anonymous union.

Program K-2

```
1  // This program demonstrates an anonymous union.
2  #include <iostream>
3  #include <iomanip>
4  using namespace std;
```

(program continues)

Program K-2 *(continued)*

```
 5
 6  int main()
 7  {
 8      union                    // Anonymous union
 9      {
10          int hours;
11          float sales;
12      };
13
14      char payType;            // To hold the pay type
15      float payRate;           // Hourly pay rate
16      float grossPay;          // Gross pay
17
18      cout << fixed << showpoint << setprecision(2);
19      cout << "This program calculates either hourly wages or\n";
20      cout << "sales commission.\n";
21
22      // Get the pay type, hourly or commission.
23      cout << "Enter H for hourly wages or C for commission: ";
24      cin >> payType;
25
26      // Determine the gross pay, depending on the pay type.
27      if (payType == 'H' || payType == 'h')
28      {
29          // This is an hourly paid employee. Get the
30          // pay rate and hours worked.
31          cout << "What is the hourly pay rate? ";
32          cin >> payRate;
33          cout << "How many hours were worked? ";
34          cin >> hours; // Anonymous union member
35
36          // Calculate and display the gross pay.
37          grossPay = hours * payRate;
38          cout << "Gross pay: $" << grossPay << endl;
39      }
40      else if (payType == 'C' || payType == 'c')
41      {
42          // This is a commission-paid employee. Get the
43          // amount of sales.
44          cout << "What are the total sales for this employee? ";
45          cin >> sales; // Anonymous union member
46
47          // Calculate and display the gross pay.
48          grossPay = sales * 0.10;
49          cout << "Gross pay: $" << grossPay << endl;
50      }
51      else
52      {
53          // The user made an invalid selection.
54          cout << payType << " is not a valid selection.\n";
55      }
56      return 0;
57  }
```

Program Output with Example Input Shown in Bold

```
This program calculates either hourly wages or
sales commission.
Enter H for hourly wages or C for commission: C Enter
What are the total sales for this employee? 12000 Enter
Gross pay: $1200.00
```

NOTE: Notice the anonymous union in Program K-2 is declared inside function `main`. If an anonymous union is declared globally (outside all functions), it must be declared static. This means the word `static` must appear before the word `union`.