Chapter 7

Defining Your Own Data Types

What Is a `struct`?

- A structure is a user-defined type
  - You define it using the keyword `struct`
  - so it is often referred as a **struct**.
- Compared to the data types we have seen, some real world objects must be described by several items:
  - Time – hh:mm:ss
  - Point – (x,y)
  - Circle – (x, y, r)

Defining a struct

```
struct POINT
{
  float x;
  float y;
};
```

- Note:
  - This doesn’t define any variables.
    - It only creates a new type.
  - Each line defining an element in the struct is terminated by a semicolon
  - A semicolon also appears after the closing brace.

Creating Variables of Type POINT

```
POINT p1, p2;
```

- If you also want to initializing a struct:

```
POINT p1 =
{
  1.0,
  2.0
};
```

Accessing the Members of a struct

- Member selection operator (.)
  - `p1.x = 3.0;`
  - `p2.y += 2.0;`

Figure 7-1 on P.334

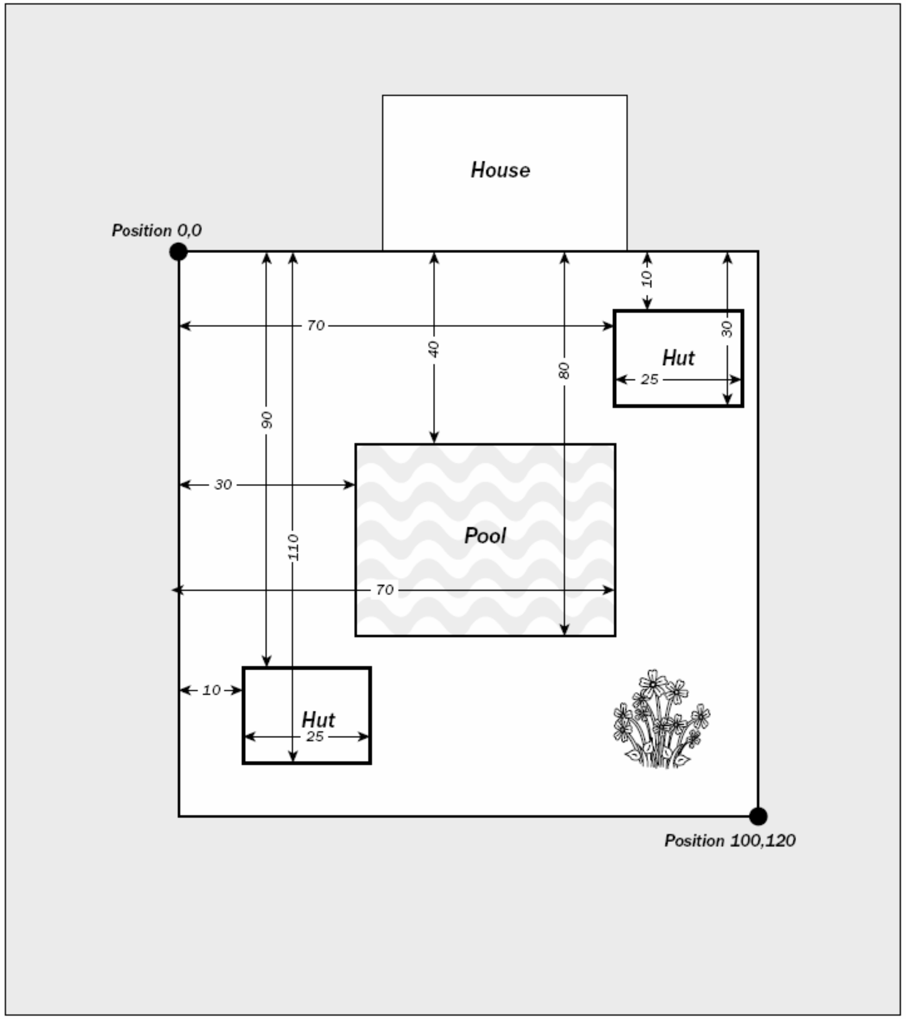
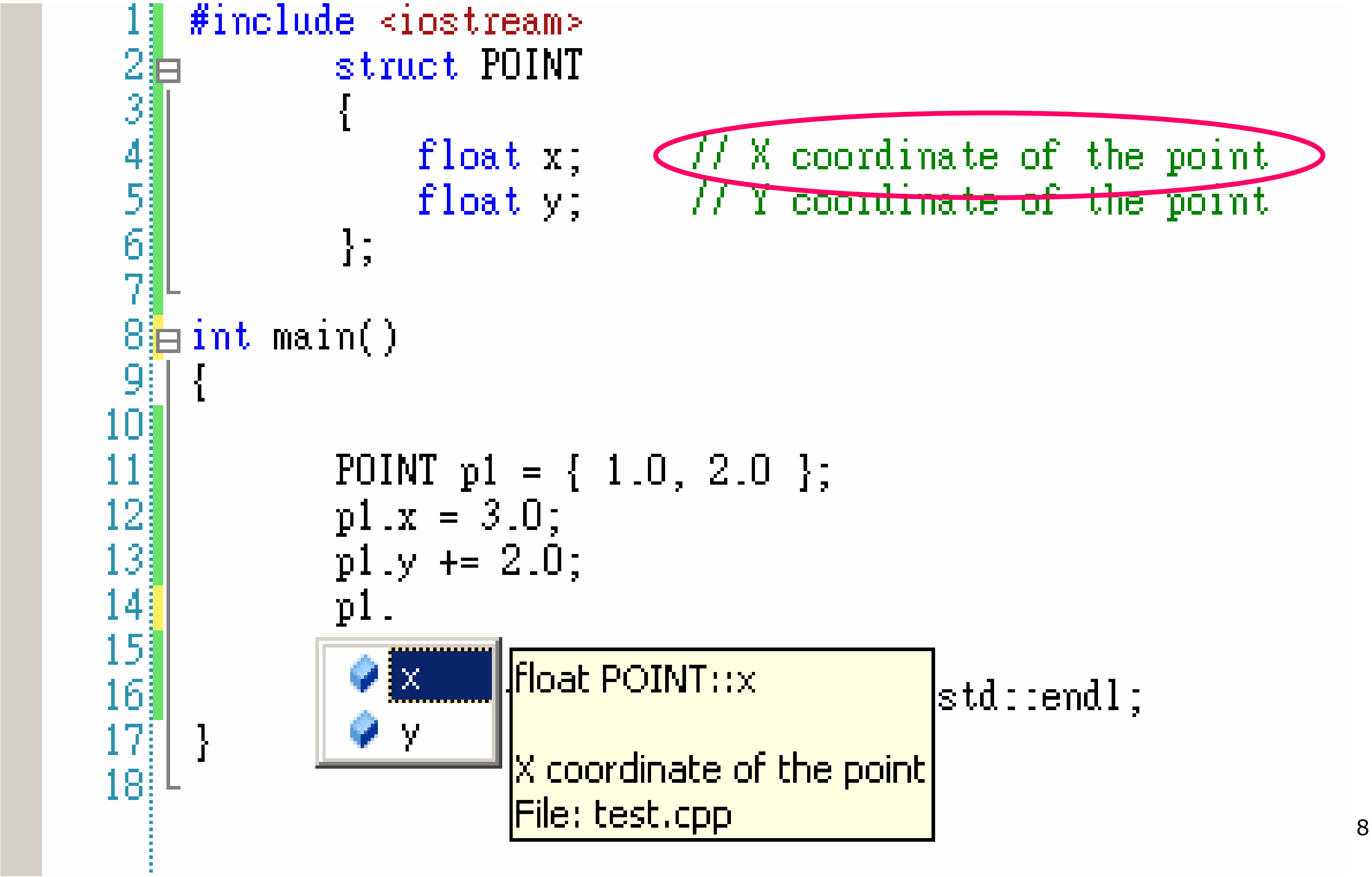


Figure 7-1

Ex7_01.cpp

- `Hut2 = Hut1;`
  - `Hut2.Left = Hut1.Left;`
  - `Hut2.Top = Hut1.Top;`
  - `Hut2.Right = Hut1.Right;`
  - `Hut2.Bottom = Hut1.Bottom;`
- Putting the definition of the struct at global scope allows you to declare a variable of type `RECTANGLE` anywhere in the `.cpp` file.
- Pass by reference

Intellisense Assistance with Structures



```
#include <iostream>
        struct POINT
        {
            float x;    // X coordinate of the point
            float y;    // Y coordinate of the point
        };

int main()
{

        POINT p1 = { 1.0, 2.0 };
        p1.x = 3.0;
        p1.y += 2.0;
        p1.

                                      std::endl;
}
```

float POINT::x

X coordinate of the point
File: test.cpp

The struct RECT

- There is a pre-defined structure RECT in the header file windows.h, because rectangles are heavily used in Windows programs.

```
struct RECT
{
  int left;              // Top left point
  int top;      // coordinate pair

  int right;             // Bottom right point
  int bottom;   // coordinate pair
};
```

Using Pointers with a struct

- `RECT* pRect = NULL;`
  - Define a pointer to RECT
- `pRect = &aRect;`
  - Set pointer to the address of aRect

A struct can contain a pointer

```
struct ListElement
{
  RECT aRect;            // RECT member of structure
  ListElement* pNext;    // Pointer to a list element
};
```

Linked List

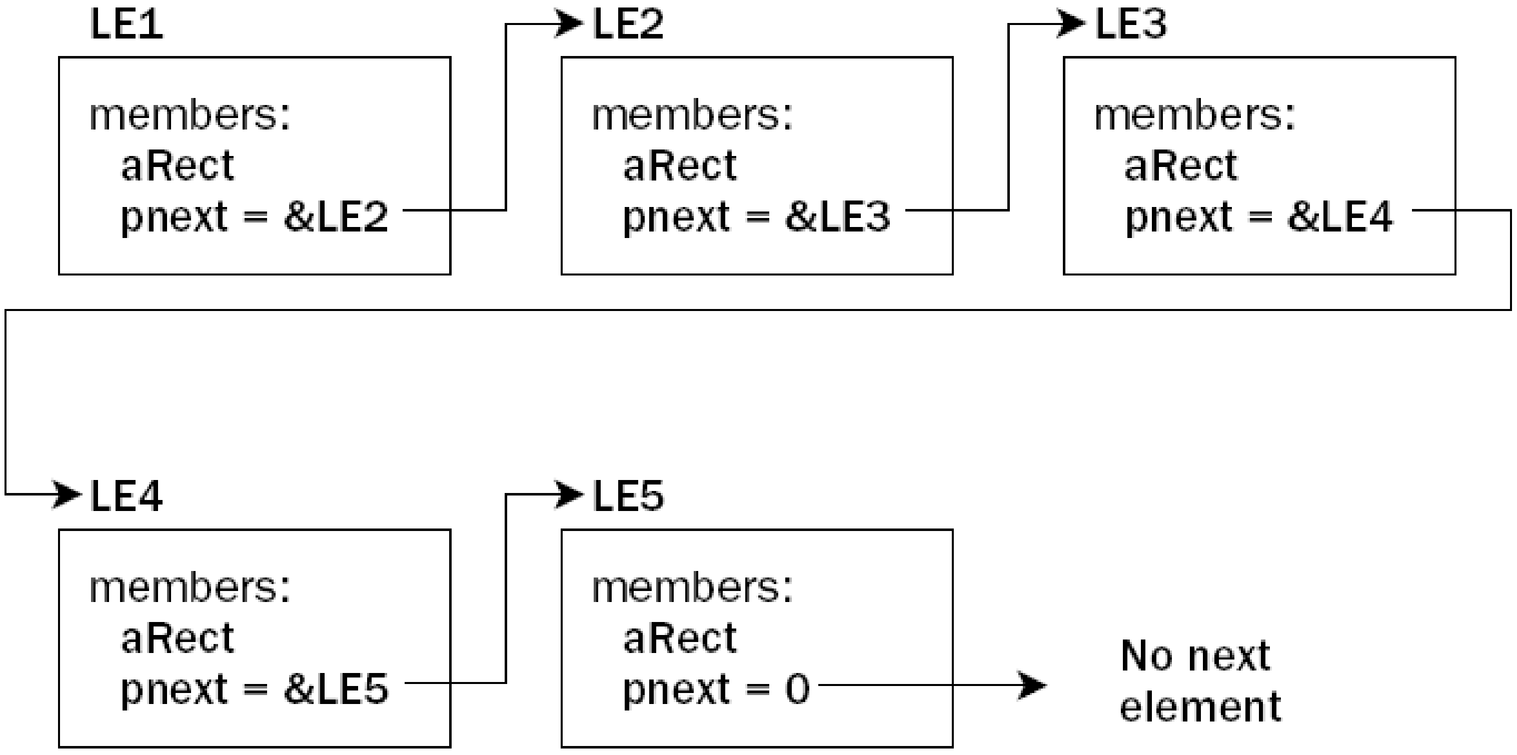


Figure 7-3

Accessing Structure Members through a Pointer

- `RECT aRect = { 0, 0, 100, 100};`
- `RECT* pRect = &aRect;`

- `(*pRect).Top += 10;`
  - The parenthesis to de-reference the pointer are necessary (P.77)

- `pRect->Top += 10;`
  - Indirect member selection operator

Exercise

- Define a struct `Sample` that contains two integer data items.
- Write a program which declares two object of type `Sample`, called `a` and `b`.
- Set values for the data items that belong to `a`, and then check that you can copy the values into `b` by simple assignment.

Dynamic Memory Allocation (P.194)

- Sometimes depending on the input data, you may allocate different amount of space for storing different types of variables at execution time

```
int n = 0;
cout << "Input the size of the vector - ";
cin >> n;
int vector[n];
```

error C2057: expected constant expression

Why Use Pointers? (P.176)

- Use pointer notation to operate on data stored in an array
- Allocate space for variables dynamically.
- Enable access within a function to arrays, that are defined outside the function

Free Store (Heap)

- To hold a string entered by the user, there is no way you can know in advance how large this string could be.

- Free Store - When your program is executed, there is unused memory in your computer.
- You can dynamically allocate space within the free store for a new variable.

The new Operator

- Request memory for a double variable, and return the address of the space
  - `double* pvalue = NULL;`
  - `pvalue = new double;`
- Initialize a variable created by new
  - `pvalue = new double(9999.0);`
- Use this pointer to reference the variable (indirection operator)
  - `*pvalue = 1234.0;`

The delete Operator

- When you no longer need the (dynamically allocated) variable, you can free up the memory space.
  - `delete pvalue;`
    - Release memory pointed to by pvalue
  - `pvalue = 0;`
    - Reset the pointer to 0

- After you release the space, the memory can be used to store a different variable later.

Allocating Memory Dynamically for Arrays

- Allocate a string of twenty characters
  - `char* pstr;`
  - `pstr = new char[20];`
  - `delete [] pstr;`
    - Note the use of square brackets to indicate that you are deleting an array.
  - `pstr = 0;`
    - Set pointer to null

Dynamic Allocation of Multidimensional Arrays

- Allocate memory for a 3x4 array
  - `double (*pbeans)[4];`
  - `pbeans = new double [3][4];`
- Allocate memory for a 5x10x10 array
  - `double (*pBigArray)[10][10];`
  - `pBigArray = new double [5][10][10];`
- You always use only one pair of square brackets following the delete operator, regardless of the dimensionality of the array.
  - `delete [] pBigArray;`

HW: Linked List

Final Exam

- Date: January 13 (Wednesday)
- Time: 14:10-17:00
- Place: TC-113

- Scope: Chapter 2-7 of Ivor Horton's Beginning Visual C++ 2008
  - CLR programming is excluded.
- Open book
- Turn off computer & mobile phone