Chapter - 12 Advanced Types

Structures

Arrays allow you to create a data collection for a single type:

```
    int data[100];    // Collection of 100 integers
```

Structures allow you to collect data of different types:

The general form of a structure definition is:

```
   variable-name;
```

Structure Usage

```
// Place for terminal cables
struct bin terminal_cable_box;
```

The *structure-name* part of the definition may be omitted.

The *variable-name* may also be omitted. This would define a structure type, but no variables.

```
};
```

Usage

Elements in a structure (called fields) are accessed by:

```
    variable.field
```

Example:

```
    // $12.95 is the new price
    printer_cable_box.cost = 1295;
```

Initialization

```
/*
 */




};



};
```

One step initialization:

```



};
```

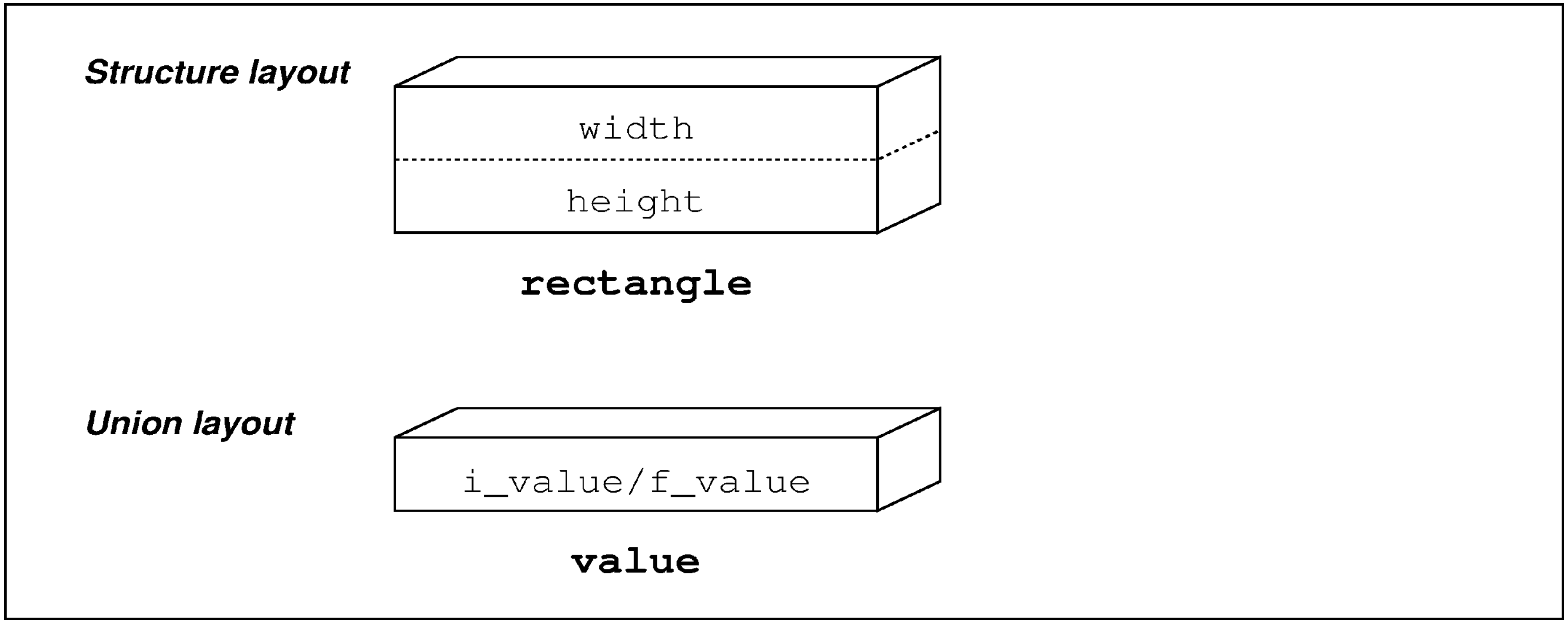
Unions

Structure -- each field is stored in a different location. Fields do not interfere with each other.

Union -- each field is stored in the same location. Changing one field puts garbage in the other fields.

```
union value {
    long int i_value; // long int version of value
    float f_value;    // floating version of value
}
```

Union Layout



Union Usage

```
/*

  */
```

[illegible]

Union Usage

```
int main(){





    data.f_value = 5.5; // store in f_value
                        //clobber i_value

    i = data.i_value;   // not legal, generates
                        // unexpected results
```

[illegible]

Union Example

```
struct circle {

};
struct rectangle {

}
struct triangle {

    int height;// Height of the triangle in pixels
};
```

Union Example

```
const int SHAPE_CIRCLE    = 0;    // Shape's circle

struct shape {
    union shape_union {// Union to hold shape info.
        struct rectangle rectangle_data;
    } data;
```

typedef

General form:

```
     typedef type-declaration.
```

The type-declaration is the same as a variable declaration except a type name is used instead of a variable name.

Example:

```
   // Define the type “width” of an object
   typedef int width;
```

We can now use our new type:

```
    width box_width;
```

Enum Type

Poor coding:

```
typedef int day_of_the_week;// define type for week days
```

Better coding: