

Problem Solving With C++

Mohammad A. Rajabi

Dept. of Geomatics Eng.

University of Tehran

Cell: 0912 132 5823

Email: marajabi@ut.ac.ir

<http://www.marajabi.com>

Structures

- Compound Data Types
- Consist of several different , yet logically connected, variables

```
struct struct-type-name {  
    type element_name1;  
    type element_name2;  
    type element_name3;  
    :  
    type element_nameN;  
};
```

Structures

```
struct inv_type {  
    char item[40]; // name of item  
    double cost; // cost  
    double retail; // retail price  
    int on_hand; // amount on hand  
    int lead_time; // number of days before  
    resupply  
};  
  
//declaring x to be of data type inv_type  
inv_type x;
```

Structures

```
inv_type x;
```

```
x.cost=10.39;
```

```
cout<<x.cost;
```

```
gets(x.item);
```

Arrays of Structures

```
inv_type invtry[100];
```

```
cout<<invtry[2].on_hand;
```

Structures (Inventory Example)

```
const int SIZE = 100;

struct inv_type {
    char item[40]; // name of item
    double cost; // cost
    double retail; // retail price
    int on_hand; // amount on hand
    int lead_time; // number of days before
    resupply
} invtry[SIZE];
```

Structures (Inventory Example)

```
void enter(), init_list(), display();
```

```
void update(), input(int i);
```

```
int menu();
```

```
int main()
```

```
{
```

```
    char choice;
```

```
    init_list();
```

```
    for(;;) {
```

```
        choice = menu();
```

```
        switch(choice) {
```

```
            case 'e': enter();
```

```
                break;
```

```
            case 'd': display();
```

```
                break;
```

```
            case 'u': update();
```

```
                break;
```

```
            case 'q': return 0;
```

```
        }
```

```
    }
```

```
}
```

Structures (Inventory Example)

```
// Initialize the inv_type_info array.
void init_list()
{
    int t;

    // a zero length name signifies
    empty
    for (t=0; t<SIZE; t++)
        *invtry[t].item = '\0';
}
```

Structures (Inventory Example)

```
// Get a menu selection.
int menu()
{
    char ch;

    cout << '\n';
    do {
        cout << "(E)nter\n";
        cout << "(D)isplay\n";
        cout << "(U)pdate\n";
        cout << "(Q)uit\n\n";
        cout << "choose one: ";
        cin >> ch;
    } while(!strchr("eduq", tolower(ch)));
    return tolower(ch);
}
```

Structures (Inventory Example)

```
// Enter items into the list.
void enter()
{
    int i;

    // find the first free structure
    for(i=0; i<SIZE; i++)
        if(!*invtry[i].item) break;

    // i will equal SIZE if the list is full
    if(i==SIZE) {
        cout << "List full.\n";
        return;
    }

    input(i);
}
} Mohammad A. Rajabi
```

Structures (Inventory Example)

```
// Input the information.
void input(int i)
{
    char str[80];
    // enter the information
    cout << "Item: ";
    cin >> invtry[i].item;

    cout << "Cost: ";
    cin >> invtry[i].cost;

    cout << "Retail price: ";
    cin >> invtry[i].retail;

    cout << "On hand: ";
    cin >> invtry[i].on_hand;
    cout << "Lead time to resupply (in days): ";
    cin >> invtry[i].lead_time;
}

```

Structures (Inventory Example)

```
// Modify an existing item.
void update()
{
    int i;
    char name[80];

    cout << "Enter item: ";
    cin >> name;

    for(i=0; i<SIZE; i++)
        if(!strcmp(name, invtry[i].item)) break;

    if(i==SIZE) {
        cout << "Item not found.\n";
        return;
    }

    cout << "Enter new information.\n";

    input(i);
}
}
}
```

Structures (Inventory Example)

```
// Display the list.
void display()
{
    int t;

    for(t=0; t<SIZE; t++) {
        if(*invtry[t].item) {
            cout << invtry[t].item << '\n';
            cout << "Cost: $" << invtry[t].cost;
            cout << "\nRetail: $";
            cout << invtry[t].retail << '\n';
            cout << "On hand: " << invtry[t].on_hand;
            cout << "\nResupply time: ";
            cout << invtry[t].lead_time << " days\n\n";
        }
    }
}
} Mohammad A. Rajabi
```

Passing Structures

```
// define a structure type
struct sample {
    int a, b;
    char ch;
} ;

void f1(sample parm);

int main()
{
    struct sample arg; // declare arg

    arg.a = 1000;

    f1(arg);

    return 0;
}

void f1(sample parm)
{
    cout << parm.a;
}
```

Assigning Structures

```
struct stype {
    int a, b;
};

int main()
{
    stype svar1, svar2;

    svar1.a = svar1.b = 10;
    svar2.a = svar2.b = 20;

    cout << "Structures before assignment.\n";
    cout << "svar1: " << svar1.a << ' ' << svar1.b;
    cout << '\n';
    cout << "svar2: " << svar2.a << ' ' << svar2.b;
    cout << "\n\n";

    svar2 = svar1; // assign structures

    cout << "Structures after assignment.\n";
    cout << "svar1: " << svar1.a << ' ' << svar1.b;
    cout << '\n';
```