Chapter 7: Structured Data

Hanoi University of Science and Technology

December 29, 2025

Slide contents are mainly based on the following documents:

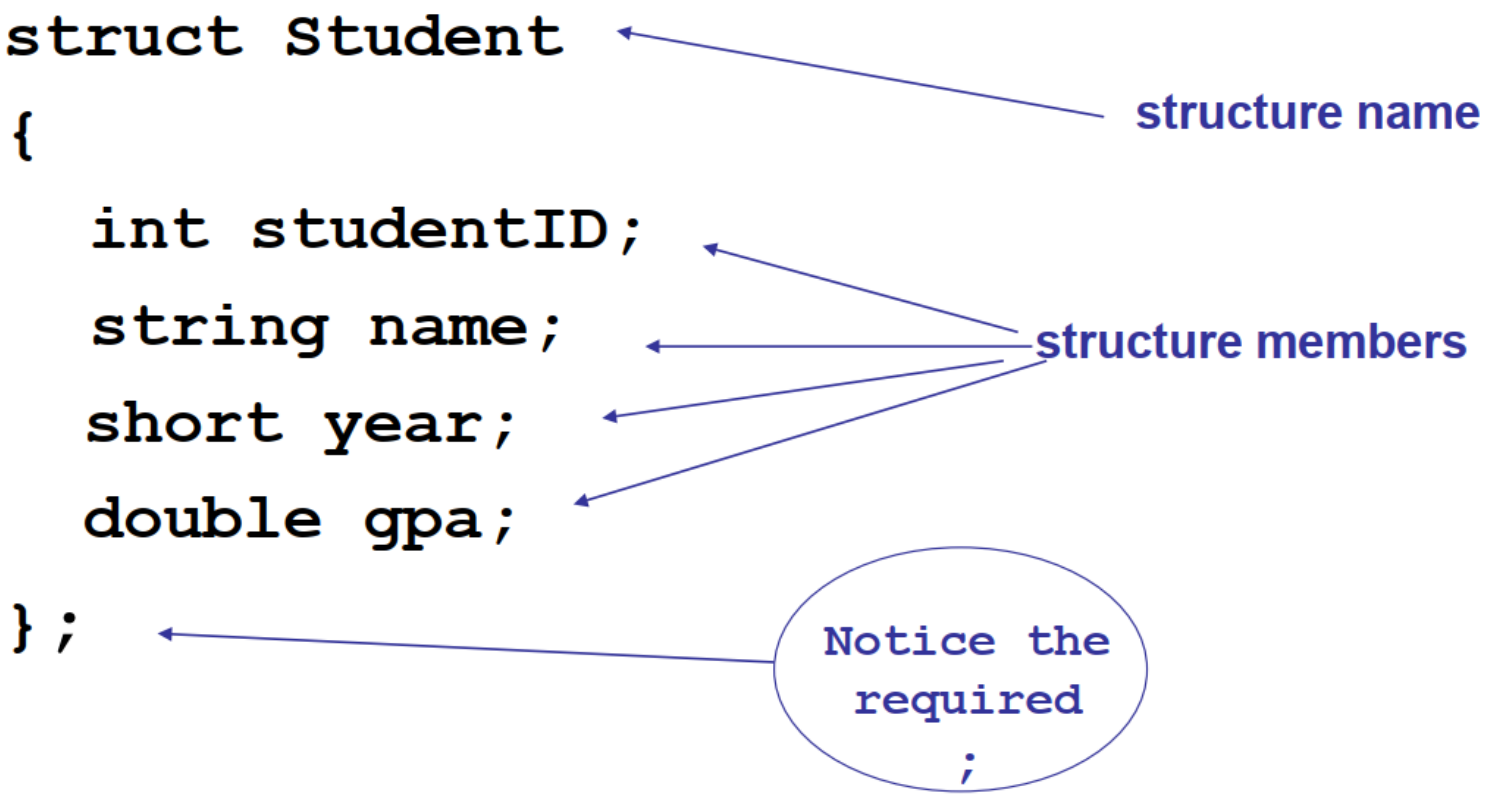
- Tony Gaddis, Starting out with C++, Six Edition, Pearson, 2009
- Tony Gaddis, Power Points Slides:
  `https://www.umsl.edu/~antognolij/gaddis.html`

Structures

- Structure: programmer-defined data type that allows multiple variables to be grouped together
- General Format:

```
struct <structName>
{
  type1 field1;
  type2 field2;
  ...
};
```

Example struct Declaration



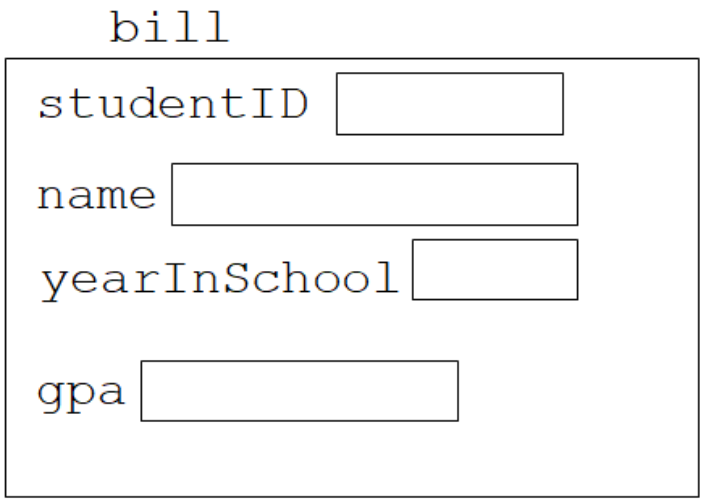
`struct` Declaration Notes

- Must have ; after closing }
- `struct` names commonly begin with uppercase letter
- Multiple fields of same type can be in comma-separated list:
  ```
  string name, address;
  ```
- Fields in a structure are all public by default

Defining Variables

- `struct` declaration does not allocate memory or create variables
- To define variables, use structure tag as type name:

```
Student bill;
```



Accessing Structure Members

- Use the dot (.) operator to refer to members of `struct` variables:

```
cin >> stu1.studentID;
getline(cin, stu1.name);
stu1.gpa = 3.75;
```

- Member variables can be used in any manner appropriate for their data type

Accessing Structure Members

Program 11-1

```
1  // This program demonstrates the use of structures.
2  #include <iostream>
3  #include <string>
4  #include <iomanip>
5  using namespace std;
6
7  struct PayRoll
8  {
9      int empNumber;     // Employee number
10     string name;       // Employee's name
11     double hours;      // Hours worked
12     double payRate;    // Hourly payRate
13     double grossPay;   // Gross pay
14 };
15
16 int main()
17 {
18     PayRoll employee; // employee is a PayRoll structure.
19
20     // Get the employee's number.
21     cout << "Enter the employee's number: ";
22     cin >> employee.empNumber;
23
24     // Get the employee's name.
25     cout << "Enter the employee's name: ";
26     cin.ignore();    // To skip the remaining '\n' character
27     getline(cin, employee.name);
28
```

Accessing Structure Members

```
29      // Get the hours worked by the employee.
30      cout << "How many hours did the employee work? ";
31      cin >> employee.hours;
32
33      // Get the employee's hourly pay rate.
34      cout << "What is the employee's hourly payRate? ";
35      cin >> employee.payRate;
36
37      // Calculate the employee's gross pay.
38      employee.grossPay = employee.hours * employee.payRate;
39
40      // Display the employee data.
41      cout << "Here is the employee's payroll data:\n";
42      cout << "Name: " << employee.name << endl;
43      cout << "Number: " << employee.empNumber << endl;
44      cout << "Hours worked: " << employee.hours << endl;
45      cout << "Hourly payRate: " << employee.payRate << endl;
46      cout << fixed << showpoint << setprecision(2);
47      cout << "Gross Pay: $" << employee.grossPay << endl;
48     return 0;
49  }
```

Accessing Structure Members

Program Output with Example Input Shown in Bold

```
Enter the employee's number: 489 [Enter]
Enter the employee's name: Jill Smith [Enter]
How many hours did the employee work? 40 [Enter]
What is the employee's hourly pay rate? 20 [Enter]
Here is the employee's payroll data:
Name: Jill Smith
Number: 489
Hours worked: 40
Hourly pay rate: 20
Gross pay: $800.00
```

Displaying a `struct` Variable

- To display the contents of a `struct` variable, must display each field separately, using the dot operator:

```
cout << bill; // won't work
cout << bill.studentID << endl;
cout << bill.name << endl;
cout << bill.yearInSchool;
cout << " " << bill.gpa;
```

Comparing `struct` Variables

- Cannot compare `struct` variables directly:

```
if (bill == william) // won't work
```

- Instead, must compare on a field basis:

```
if (bill.studentID == william.studentID)
   ...
```

Initializing a Structure

- `struct` variable can be initialized when defined:

```
Student s = {11465, "Joan", 2, 3.75};
```

- Can also be initialized member-by-member after definition:

```
s.name = "Joan";
s.gpa = 3.75;
```

More on Initializing a Structure

- May initialize only some members:
  `Student bill = {14579};`
- Cannot skip over members:
  `Student s = {1234, "John", , 2.83}; // illegal`
- Cannot initialize in the structure declaration, since this does not allocate memory

```
struct Student      // Illegal
{                   // initialization
  int studentID = 1145;
  string name = "Alex";
  short year = 1;
  float gpa = 2.95;
};
```

Arrays of Structures

- Structures can be defined in arrays
- Can be used in place of parallel arrays

```
const int NUM_STUDENTS = 20;
Student stuList[NUM_STUDENTS];
```

- Individual structures accessible using subscript notation
- Fields within structures accessible using dot notation:

```
cout << stuList[5].studentID;
```

Arrays of Structures

Program 11-4

```
1  // This program uses an array of structures.
2  #include <iostream>
3  #include <iomanip>
4  using namespace std;
5
6  struct PayInfo
7  {
8       int hours;        // Hours worked
9       double payRate;   // Hourly pay rate
10 };
11
12 int main()
13 {
14      const int NUM_WORKERS = 3;    // Number of workers
15      PayInfo workers[NUM_WORKERS]; // Array of structures
16      int index;                    // Loop counter
17
18      // Get employee pay data.
19      cout << "Enter the hours worked by " << NUM_WORKERS
20           << " employees and their hourly rates.\n";
21
```

Arrays of Structures

```
22      for (index = 0; index < NUM_WORKERS; index++)
23      {
24          // Get the hours worked by an employee.
25          cout << "Hours worked by employee #" << (index + 1);
26          cout << ": ";
27          cin >> workers[index].hours;
28
29          // Get the employee's hourly pay rate.
30          cout << "Hourly pay rate for employee #";
31          cout << (index + 1) << ": ";
32          cin >> workers[index].payRate;
33          cout << endl;
34      }
35
36      // Display each employee's gross pay.
37      cout << "Here is the gross pay for each employee:\n";
38      cout << fixed << showpoint << setprecision(2);
39      for (index = 0; index < NUM_WORKERS; index++)
40      {
41          double gross;
42          gross = workers[index].hours * workers[index].payRate;
43          cout << "Employee #" << (index + 1);
44          cout << ": $" << gross << endl;
45      }
46      return 0;
47  }
```

Arrays of Structures

Program Output with Example Input Shown in Bold

```
Enter the hours worked by 3 employees and their hourly rates.
Hours worked by employee #1: 10 [Enter]
Hourly pay rate for employee #1: 9.75 [Enter]

Hours worked by employee #2: 20 [Enter]
Hourly pay rate for employee #2: 10.00 [Enter]

Hours worked by employee #3: 40 [Enter]
Hourly pay rate for employee #3: 20.00 [Enter]

Here is the gross pay for each employee:
Employee #1: $97.50
Employee #2: $200.00
Employee #3: $800.00
```

Nested Structures

- A structure can contain another structure as a member:

```
struct PersonInfo
{  string name,
           address,
           city;
};
struct Student
{   int studentID;
    PersonInfo pData;
    short yearInSchool;
    double gpa;
};
```

Members of Nested Structures

- Use the dot operator multiple times to refer to fields of nested structures:

```
Student s;
s.pData.name = "Joanne";
s.pData.city = "Tulsa";
```

Structures as Function Arguments

- May pass members of `struct` variables to functions:
  ```
  computeGPA(stu.gpa);
  ```
- May pass entire struct variables to functions:
  ```
  showData(stu);
  ```
- Can use reference parameter if function needs to modify contents of structure variable
- If structure data should not be changed, use a const reference parameter:
  ```
  void showData(const Student &s) // header
  ```

Returning a Structure from a Function

- Function can return a `struct`:

```
Student getStuData(); // prototype
stu1 = getStuData(); // call
```

- Function must define a local structure:
  - for internal use
  - for use with `return` statement

Returning a Structure from a Function

```
Student getStuData()
{ Student s;     // local variable
  cin >> s.studentID;
  cin.ignore();
  getline(cin, s.pData.name);
  getline(cin, s.pData.address);
  getline(cin, s.pData.city);
  cin >> s.year;
  cin >> s.gpa;
  return s;
}
```

Returning a Structure from a Function

- Programming Challenges: 1-7
- Define a fraction structure type consisting of a numerator and a denominator. Then, write functions for:
  - simplification of a fraction
  - addition of 2 fractions
  - subtraction of 2 fractions
  - multiplication of 2 fractions
  - division of 2 fractions