Structured Data

1. Abstract Data Types

You have so far used *primitive data types* which are data types (such as int, float, char, bool) that are built-in to the language.

Abstract data types, also referred to as *user-defined data types*, are data types created by the programmer to represent something.

Abstraction is the process of taking away or removing characteristics from something in order to reduce it to a minimal set of essential characteristics.

For example, if you want to store the height of a person, it is not enough to declare a variable of type int to store just a number. There are at least two problems with this: 1) if you use feet and inches then you need to have two numbers, one for feet and one for inches; and 2) may be you want to have the option to use either the metric system or the English system, in which case you need to qualify the number with a unit.

Let's say you want to just use the English system to store the height in feet and inches. You can declare two int variables called feet and inches, but in doing this, there's nothing that suggests that these two numbers must be taken together to represent one thing – the height. So what we really want to do is to define our own data type for storing the height that has two int's.

2. Structures

C++ allows you to group several variables together into a single item known as a structure. Here's the structure syntax for creating a user-defined data type:

```
struct type-name {
    variable declaration;
    variable declaration;
    // as many as you need
};
```

All the variables declared inside the structure are known as *data members*. Here's an example for creating the Height user-defined data type with two data members:

```
struct Height {
    int feet;
    int inches;
};
```

It's important to note that the structure declaration does not define a variable for storing the actual data. It simply tells the computer what the structure is made of. In essence, it creates a new user-defined data type of that given name. To use this new data type, you need to declare a

 Copyright Enoch Hwang 2023

variable of this new data type, just as you would with any other variables and data types. For example, the following declares a variable called myHeight of type Height.

```
Height myHeight;
```

3. Accessing Data Members

To access the data members inside a structure, we start with the name of the variable for that structure followed by the dot operator (.) and then the variable name of the data member. For example, the following assigns values to the two data members feet and inches in the variable myHeight.

```
myHeight.feet = 5;
myHeight.inches = 4;
```

4. Comparing Structure Variables

You cannot compare the entire structure variables like this

```
Height myHeight, yourHeight;
if (myHeight == yourHeight) …
```

Instead you need to compare the individual data members like this

```
if (myHeight.feet == yourHeight.feet && myHeight.inches == yourHeight.inches) …
```

 Copyright Enoch Hwang 2023

5. Example 1

To store the height for a person

```
#include <string>

int main() {
  struct Height {
    int feet;
    int inches;
  };

  struct Person {
    string name;
    Height height;
  };

  Person person;

  person.name = "Dr Hwang";
  person.height.feet = 5;
  person.height.inches = 4;
}
```

6. Example 2

To store the height for 10 people

```
#include <string>

int main() {
  struct Height {
    int feet;
    int inches;
  };

  struct Person {
    string name;
    Height height;
  };

  Person person[10];

  person[0].name = "Dr Hwang";
  person[0].height.feet = 5;
  person[0].height.inches = 4;
}
```

 Copyright Enoch Hwang 2023

7. *Exercises* (Problems with an asterisk are more difficult)

1. Modify the Person's structure to allow it to store also the person's birthday, and address.

2. Write sample code to assign values into a Person's variable, and then print the data out.

3. Create a variable that can store 25 people's data using the Person data type from question 1.

4. Write sample code to assign values into the variable declared in question 3, and then print the data out.

5. * I operate 25 weather stations around the world.
   I want to store the temperatures obtained from these weather stations on a daily basis.
   I want to keep a history of these daily temperature readings for 10 days for each weather station.

   Design a data structure for storing these data. Data to store:

   - Name of station
   - Location of station: longitude and latitude
   - Temperature: degree and scale

   Write a program to implement this data object. Assign sample data for one station.

 Copyright Enoch Hwang 2023