



CS250 Intro to CS II

Spring 2014

Chapter 11

Structured Data

- Reading: pp. 593-608, 612-617
- Good Problems to Work: p. 604 11.1; p 611 11.4, 11.5, 11.6, 11.7; p. 641 34

Primitive Data Types

- The primitive data types (defined as part of the language) are:

`bool, char, unsigned char, short int, int
long int, unsigned short int, unsigned int,
unsigned long int, float, double, long
double`

Programmer-defined Data Types or Abstract Data Types (ADTs)

- ADTs are data types created by the programmer with their own domain/range and operations.
- ADTs are composed of one or more primitive data types.

Enumerated Data Types are ADTs

- An enumerated data type is a programmer-defined data type

General Format

```
enum TypeName {One or more enumerators};
```

Example

```
enum Day {MON, TUE, WED, THU, FRI, SAT, SUN};  
Day day;  
day = MON;
```

- The enumerators are integer constants the compiler assigns starting with 0 unless otherwise specified

Enumerated Data Types

```
Day day;  
int whatDay, indx;  
day = 3;           // illegal  
  
whatDay = TUE;    // legal  
  
if (day > WED)   // legal  
  
for (indx = MON; indx <= SUN; ++indx) // legal  
day = static_cast<Day> (day + 1); // legal
```

Enumerated Data Types

```
switch (day)
{
    case MON:          cout << "Monday";
                      break;

    case TUE:          cout << "Tuesday";
                      break;

    ...

}
```

- Anonymous Enumerator Data Types

```
enum {FREEZING = 32, BOILING = 212};
```

Structures

- A struct (structure) is another example of a programmer-defined data type that can be used to declare variables

```
struct Time // declaration of a struct called
Time
{
    int mHours,
        mMinutes,
        mSeconds;
}; // notice the ; is mandatory
```


Problem

- Create a variable of type Time and initialize the time to 1:30pm
- Answer:

```
Time sTime; // notice s prefix for variables  
  
// The . operator allows access to structure  
// members  
sTime.mHours = 13;  
sTime.mMinutes = 30;  
sTime.mSeconds = 0;
```

struct Initialization

- Here is another way to initialize members of a struct

```
Time sTime1 = {13, 30, 0}; // legal
```

```
Time sTime2 = {13, 30}; // seconds undefined
```

```
Time sTime3 = {13, , 0}; // illegal
```

Operations on structs

- Which of the following C++ statements are legal given variables sTime1 and sTime2 are of type Time?

a) `cout << sTime1 << sTime2;`

b) `if (sTime1 == sTime2)`
`{`
 `cout << "times are equal";`
`}`

c) `cout << sTime1.mHours;`

d) `cin >> sTime1;`

e) `cin >> sTime1.mHours;`

f) `sTime1 = sTime2;`

structs as Function Arguments

- Write a function `printTime` that accepts a `Time` and prints the time in the form `xx:xx:xx` so `1:30` would be `01:30:00`

```
void printTime (Time sTime)
{
    cout << setfill ('0') << setw (2) << sTime.mHour << ':' <<
        << setw (2) << sTime.mMinute << ':' <<
        << setw (2) << sTime.mSecond << endl;
}
```

- What happens if we change

```
void printTime (Time sTime) to
```

```
void printTime (const Time &sTime)
```

Arrays of Structures

- Consider the following struct

```
const int MAX_STRING = 64;
struct BookInfo
{
    char mTitle[MAX_STRING];
    char mAuthor[MAX_STRING];
    char mPublisher[MAX_STRING];
    double mPrice;
};
```

1. Declare an array that can hold 1000 books
2. Write a function **printBookNames** that will print the names of the books with a price under \$50