• Structured Data types
• Data abstraction
• structs
• array of structs
Data Types in C++

simple

integral
- int
- char
- short
- long
- [ unsigned ]
- bool
- enum

floating
- float
- double
- long double

address
- pointer
- reference

structured
- array
- class
- struct
- union
Structured Data Types

• simple data type
  – atomic values
    '%' 18 -1.0 true

• structured type (a.k.a. composite)
  – has component parts and organized structure
  – a collection or group of elements
Why Structured Types?

- Group together multiple simple items in a single object.

- Examples:
  - Entries in the campus directory
    - Last Name
    - First Name
    - Phone Number
    - Address
    - Major
  - FootBall Game
    - season
    - opponent
    - location
    - win/lose/tie
    - UM score
    - opponent score
Picturing a single game

<table>
<thead>
<tr>
<th>Season</th>
<th>2006</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opponent</td>
<td>&quot;Vanderbilt&quot;</td>
</tr>
<tr>
<td>Location</td>
<td>Ann Arbor, MI</td>
</tr>
<tr>
<td>Win/Lose/Tie</td>
<td>W</td>
</tr>
<tr>
<td>UM Score</td>
<td>27</td>
</tr>
<tr>
<td>Opponent Score</td>
<td>7</td>
</tr>
</tbody>
</table>

game1
struct Declaration: Syntax

struct <Type Name> 
{
    <Data Type> <Member Name>;
    <Data Type> <Member Name>;
    . . .
};


**struct (ADT record)**

- A **struct** lets you put related data together of mixed data types

```c
struct FootballType {
    int season;
    string opponent;
    string location;
    char W_L_T;
    int UM_score, opponent_score;
};
```
Review: Using structs

- Declared like simple types:

  ```
  FootballType game1, game2;
  ```

- Access components of a struct using the “dot operator”, for example

  ```
  game1.season = 2006;
  game1.opponent = "Valderbilt";
  ```
Member Selector

Dot operation

<Struct Variable>.<Member Name>

used in expressions to access member components
Picturing a single football record

2006
"Vanderbilt"
Ann Arbor, MI
W
27
7

game1
Aggregate Operations on structs

- I/O: NO
- Assignment: YES
- Arithmetic: NO
- Comparison: NO
- Pass as Argument: YES
  - by value or by reference
- Return as function result: YES
Can use struct members just like any other variable of that type

- // add one to game1's UM_score
  game1.UM_score++;

- // print a formatted title
  cout << setw(50) << game1.opponent;

- // assign game 1 to game2
- // copies ALL members
  game2 = game1;
Passing a struct by Value

// example call from main
// printOneGame (game1);

void printOneGame (Game game) {
    cout << "Season: " << game.season << endl
       << "Opponent: " << game.opponent << endl
       << "Score: "
       << game.UM_score << " "
       << game.opponent_score << endl;
    if (game.W_L_T == 'W')
        cout << "UM ";
    else if (game.W_L_T == 'L')
        cout << "game.opponent"
    else cout << "tie";
}

Passing a struct by Reference

// example call from main
// getScores (game1);

void getScores (Game& thisGame)
{
    cout << "Enter UM Score: " ;
    cin >> score;
    thisGame.UM_score = score;
    cout << "Enter opponent score: " ;
    cin >> score;
    thisGame.opponent_score = score;
}
Nested (hierarchical) structs

- Members of a struct may be structs
- When to nest structs
  - if nesting improves readability
  - if a structure is used repeatedly inside other structs
  - Beware: too many levels of nesting may impair readability
Example: nested structs

// add date to our game info

struct DateRec // declare a type to
    // represent a date
{
    int month,
        day,
        year;
};
struct NewGameRec
{
    int season;
    DateRec datePlayed;
    string opponent, location
    char W_L_T;
    int UM_score;
    int opponent_score;
} game3;

Declarations allowed here
How can we picture this new game?

```
game3

season  ←
month

day  ←
year

opponent
location
W_L_T
UM_score
opponent_score

 game3.datePlayed
```

- `game3.season`
- `game3.dataPlayed.month`
- `game3.dataPlayed.day`
- `game3.dataPlayed.year`
Data Structures

• What is a data structure?
  – Layout of information in a program to hold whatever data it needs
    • Homogeneous data layout (arrays)
    • Heterogeneous data layout (structures)
    • Combinations of the two

• Choosing a data structure is an important step in writing a complex program…
Arrays of Structs

• Example: for the game of an entire season; declaring separate variables like `game1` and `game2` is not practical

• There are many options for a suitable data structure; in this course (with what we know) an *array of structs* is a good choice
// first, declare the struct type as before

struct FootballType
{
    int season;
    string opponent;
    string location;
    char W_L_T;
    int UM_score, opponent_score;
};

Declaring the basetype of the array
Declaring the array of structs

// declare a constant for the # of games
const int NUMGAMES = 100;

// declare the actual array variable
FootballType games[NUMGAMES];

one-dimensional array
has 100 elements
each element is a
FootballType
the array

- games [0]
- •
- •
- •
- games [99]
- games
one 'FootBallType'

- season
- opponent
- location
- W_L_T
- UM_score
- opponent_score
the array

games [0]

season
opponent
location
W_L_T
UM_score
opponent_score

games [1]

season
opponent
location
W_L_T
UM_score
opponent_score

games
where every games[i] is this:

- games[i].season
- games[i].opponent
- games[i].location
- games[i].W_L_T
- games[i].UM_score
- games[i].opponent_score

games[i]
Accessing this array

- entire array  
  type array of FootballType

  games

- entire game record at entry i

  games[i]  
  type FootballType

- opponent of game at entry i

  games[i].opponent  
  type string

- char in the jth position of the opponent of the game at index i

  games[i].opponent[j]  
  type char
Passing games and game structs

- If pass entire games array, follow rules for passing an ARRAY
  - arrays are ALWAYS passed by reference
  - use `const` reference to block changes
  - example call:
    ```
    processAllGames ( games );
    ```

array of FootballType
Prototypes

• Example prototypes
  – array is input only to the function
    ```cpp
    void processAllGames (const FootballType games[]);
    ```

  – array is modified by the function
    ```cpp
    void processAllGames ( FootballType games[] );
    ```
Calling with an element of the array

```
processOneGame ( games[i] );
```

- If pass a single game record, follow rules for passing a STRUCT
  - pass by value or by reference
Data Structures

• What is a data structure?
  – Layout of information in a program to hold whatever data it needs
    • Homogeneous data layout (arrays)
    • Heterogeneous data layout (structures)
    • Combinations of the two

• Choosing a data structure is an important step in writing a complex program…
Pick a Data Structure to hold...

- ...all of the states in the USA
- ...the season statistical totals for a football player
- ...the service information for a car
- ...the daily price of a stock for the last 6 months
- ...the ID numbers of passengers on a plane
- ...the number of stars in each square degree of the sky
- ...the directory information for a University student
- ...the directory information for all University students