# Computer Science 5C
# Chapter 7--Enumeration Types and Typedef

Dr. Scott A. Brandt
*Professor, Computer Science Department*
*Director, UCSC/LANL Institute for Scalable Scientific Data Management*

# Enumeration types

- The keyword *enum* is used to declare enumeration types
- Enumeration types (often called enums) are new, user-defined types
- Enums are named sets of named values
- The type and the values must all be valid identifiers
- Example
  - enum day {sun, mon, tue, wed, thu, fri, sat};
  - enum day d1, d2;  /* enum day is the type */
  - d1 = fri;

# More examples

1. enum suit {clubs, diamonds, hearts, spades};
2. enum suit a, b, c;
3. enum count {one, two, three} x, y;
4. enum count {one=1, two, three} x, y;
5. enum fruit {apple=7, pear, orange=3, lemon};
6. enum veg {beet=17, corn=17} vege1, vege2;
7. enum {fir, pine, oak, elm} tree;

# Typedef

- The *typedef* keyword can be used to name a type
  - The name must be a valid identifier
- Example
  - typedef int color;
  - color red, blue, green;  /* these are really ints */
  - red = 1;  blue = 5;
  - if(red == green)  /* etc. */

# typedef and enum

- Typedef can provide a new name for any type, including enumeration types
- Example
  - enum day {sun, mon, tue, wed, thu, fri, sat};
  - enum day d1;
  - typedef  enum day  day;
  - day d2;

# Using enums

- enum day {sun, mon, tue, wed, thu, fri, sat};
- typedef enum day day;
- day find_next_day(day d) {

```
    if(day == sat)
        return sun;
    else
        return (day) ( (int)day + 1 );
}
```

- day find_next_day(day d)

```
    day next_day;

    switch(d) {
      case sun:
        next_day = mon;
        break;
      case mon:
        next_day = tue;
        break;
      /* etc. */
    }
    return next_day;
  }
```

# Rock, scissors, paper

- Rock breaks scissors
- Scissor cuts paper
- Paper covers rock

# prs.h

```
#include <ctype.h>     /* for isspace() */
#include <stdio.h>     /* for printf(), etc. */
#include <stdlib.h>    /* for rand() and srand() */
#include <time.h>      /* for time() */

enum prs {paper, rock, scissors, game, help, instructions, quit};

enum outcome {win, lose, tie, error};

typedef enum prs prs;
typedef enum outcome outcome;
```

# prs.h (continued)

```
outcome compare(prs player_choice, prs
computer_choice);
void prn_final_status(int win_cnt, int lose_cnt);
void prn_game_status(int win_cnt, int lose_cnt, int tie_cnt);

void prn_help(void);
void prn_instructions(void);
void report(outcome result, int *win_cnt_ptr,
                int *lose_cnt_ptr, int *tie_cnt_ptr);
prs computer_selection(void);
prs player_selection(void);
```

# main.c

```
#include <prs.h>

int main(void)
{
    int win_cnt = 0, lose_cnt = 0, tie_cnt = 0;
    outcome result;
    prs player_choice, computer_choice;

    srand(time(NULL));
    prn_instructions();

    while((player_choice = player_selection()) != quit)
        switch(player_choice) {
            case paper:
            case rock:
            case scissors:
            machine_choice = machine_selection();
```

# main.c (continued)

```
            result = compare(player_choice, machine_choice);
            report(result, &win_cnt, &lose_cnt, &tie_cnt);
            break;
        case game:
            prn_game_status(win_cnt, lose_cnt, tie_cnt);
            break;
        case instructions:
            prn_instructions();
            break;
        case help:
            prn_help();
            break;
        default:
            printf("ERROR");
            exit(1);
    }
    prn_game_status(win_cnt, lose_cnt, tie_cnt);
    prn_final_status(win_cnt, lose_cnt);
}
```

# prn.c

```
#include <prs.h>

void prn_final_status(int win_cnt, int lose_cnt)
{
    if(win_cnt > lose_cnt)
        printf("Congratulations -- you won!\n");
    else if(win_cnt == lose_cnt)
        printf("TIE\n");
    else
        printf("Sorry -- you lost!\n");
}
```

# prn.c (continued)

```
void prn_game_status(int win_cnt, int lose_cnt, int
tie_cnt)
{
    printf("\nGAME STATUS: \n");
    printf("    Win:    %4d\n", win_cnt);
    printf("   Lose:    %4d\n", lose_cnt);
    printf("    Tie:    %4d\n", tie_cnt);
    printf("  Total:    %4d\n", win_cnt+lose_cnt
+tie_cnt);
}
```

# prn.c (continued)

```
void prn_help(void)
{
    printf("\n%s\n%s\n%s\n%s\n%s\n%s\n%s\n",
    " p  for paper",
    " r  for rock",
    " s  for scissors",
    " g  for game status",
    " h  for help",
    " i  for instructions",
    "  q  to quit the game");
}
void prn_instructions(void) {
    prn_help();
}
```

# select.c

```
#include <prs.h>

prs machine_selection(void)
{
    return ((prs)(rand()%3));
}

prs player_selection(void)
{
    char c;
    prs player_choice;

    printf("Input p, r, or s:");
    scanf(" %c", &c);
```

# select.c (continued)

```
    switch(c) {
      case ‘p’:
        player_choice = paper;
        break;
      case ‘r’:
        player_choice = rock;
        break;
      case ‘s’:
        player_choice = scissors;
        break;

     /* etc. */
    }
    return player_choice;
};
```

# compare.c

```
#include <prs.h>

outcome compare(prs player_choice, prs computer_choice)
{
    outcome result;

    if(player_choice == computer_choice)
        return tie;
    switch(player_choice) {
        case paper:
          result = (machine_choice == rock) ? win : lose;  break;
        case rock:
          result = (machine_choice == scissors) ? win : lose;  break;
       case scissors:
         result = (machine_choice == paper) ? win : lose;  break;
    }
    return result;
}
```

# report.c

```
void report(outcome result, int *win_cnt_ptr, int *lose_cnt_ptr, int
*tie_cnt_ptr)
{
    switch(result) {
      case win:
        *win_cnt_ptr++;
        break;
      case lose:
        *lose_cnt_ptr++;
        break;
      case tie:
        *tie_cnt_ptr++;
        break;
    }
}
```