C Programming Language

(6th class)

Dohyung Kim

Assistant Professor @ Department of Computer Science

Today …

- Let’s practice programing!
  - Tic-Tac-Toe

Tic-tac-toe

- **Tic-tac-toe** (also known as noughts and crosses or Xs and Os) is a **paper-and-pencil game** for two players, *X* and *O*, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game.

What should we do first?

- **Try to design a program**
  - Initialize (maybe setting up the table, prepare markers)
  - Get user info (simple)
  - Display table
  - Aha… we may need a loop for each player's turn.
    - Take user input (how to take user inputs and store them)
    - Update the table and display it (simple)
    - Check condition (should we continue or end?)
    - Change players
  - …

Initialization

- **We have a 3 by 3 sized table.**
  - What can you use to represent/store the table?

Initialization

- **We have a 3 by 3 sized table.**
  - Array? 2-dimensional?
  - Array (char table[3][3]) could represent the status of the table.

```
char table[3][3] = {{'_', '_', '_'}, {'_', '_', '_'}, {'_', '_', '_'}};
```

  - Please check whether you can see 3x3 table on the screen (print out table)

Initialization

- **We have a 3 by 3 sized table.**
  - Array? 2-dimensional?
  - Array (char table[3][3]) could represent the status of the table.

```
char table[3][3] = {{'_', '_', '_'}, {'_', '_', '_'}, {'_', '_', '_'}};
```

  - Please check whether you can see 3x3 table on the screen (print out table)

- **Markers for two player**
  - char markers[2] = {'o', 'x'};

Get User Info

- **Get the player's name**
  - You may already know how to get it
  - Print a string and get the user input by using **scanf**

Get User Info

- **Get the player's name**
  - You may already know how to get it
  - Print a string and get the user input by using **scanf**

```
char p[2][20];
printf("\n Enter the name of Player 0:");
scanf("%s", p[0]);
printf("\n Enter the name of Player 1:");
scanf("%s", p[1]);
```

Display the table

- **How to print 3 by 3 sized array?**

Display the table

- **How to print 3 by 3 sized array?**
  - Yeah, that's it. **For loop**

Display the table

- **How to print 3 by 3 sized array?**
  - Yeah, that's it. **For loop**

```
int i, j;
for (i=0; i<3; ++i) {
   for (j=0; j<3; ++j){
      printf("%c ", table[i][j]);
   }
   printf("\n");  /* to separate each row */
}
```

Loop

- do while?
- while (1) ?
- whatever you want

Take user input

- **At each turn, you are supposed to take position info on the table from users.**

Take user input

- **At each turn, you are supposed to take position info on the table from users.**
  - Print a message and get the user input by using **scanf**

Take user input

- **At each turn, you are supposed to take position info on the table from users.**
  - Print a message and get the user input by using **scanf**

```
if (turn == 0){
   printf("P0's turn. Input location (row, col) : ");
}else{
   printf("P1's turn. Input location (row, col) : ");
}

scanf ("%d %d", &row, &col);
```

Take user input

- **At each turn, you are supposed to take position info on the table from users.**
  - Print a message and get the user input by using **scanf**

```
if (turn == 0){
   printf("P0's turn. Input location (row, col) : ");
}else{
   printf("P1's turn. Input location (row, col) : ");
}

scanf ("%d %d", &row, &col);
```

  - **'turn' indicates which player is now playing**
    - **0 if player 0 is playing, 1 otherwise.**

Update the table and display it

- **Update?**
  - Set the table element of the position that a player specified.

Update the table and display it

- **Update?**
  - Set the table element of the position that a player specified

- **Display it?**
  - The same for loop.
  - We'd better make a display module as a function

Check condition

- **We need to check each row, column, and diagonal.**
- **How?**
  - We can compare characters

```
char m;
int done = 0;
int cnt = 0, i, j;

/* check each row */
for (i=0, i<3; ++i){
   m = table[i][0];
   if(done == 1)
      break;
   for (j=0; j<3; j++){
      if(m == table[i][j]){
         ++cnt;
      }
      if(cnt == 3){
        done = 1;
      }
      if(j == 3){
         cnt = 0;
      }
 }
```

Check condition

- **We need to check each row, column, and diagonal.**
- **How?**
  - We can compare characters

```
char m;
int done = 0;
int cnt = 0, i, j;

/* check each row */
for (i=0, i<3; ++i){
   m = table[i][0];
   if(done == 1)
      break;
   for (j=0; j<3; j++){
      if(m == table[i][j]){
         ++cnt;
      }
      if(cnt == 3){
        done = 1;
      }
      if(j == 3){
         cnt = 0;
      }
}
```

Complicate!

- **We need to check each row, column, and diagonal.**
- **How?**
  - We can check the terminating conditions using numbers assigned to each character

```
int made[2] = {3*'o', 3*'x'};

/* check each row */
for (i = 0; i<3; ++i){
   sum = 0;
   for (j = 0; j<3; ++j){
      if(table[i][j] == '_')
         break;
      sum += table[i][j];
   }
   if (sum == made[0] or sum == made[1])
      done = 1;
   if (done)
      break;
}
```

Change players

- By changing the value of ‘turn’

Change players

- **By changing the value of ‘turn’**

```
++ turn;
turn = turn % 2;
```

Need more

- Check occupation (what if a player tries to set its mark in the position that has been already occupied)
- Error handling (what if players give the position indices which are not valid in the table (e.g., 1, 3 3, x 1))
- …

Q and A