دانشکده علوم ریاضی

گروه علوم کامپیوتر

آموزش گام‌به‌گام برنامه‌نویسی زبان C

استاد درس:

خیام صالحی

تهیه و تنظیم:

کارو صحافی

karosahafi@gmail.com

پاییز ۱۳۹۳

به نام خدا

در این مقاله کوتاه می‌خواهیم بازی ساده دوز را در زبان برنامه‌نویسی C پیاده‌سازی کنیم و با اشتباهات رایج دانشجویان در برنامه‌نویسی آشنا شویم. همیشه قبل از شروع نوشتن یک پروژه باید مسیر خود را مشخص کرده و پس از انجام هرگام، گام بعدی ما چه خواهد بود. نکته مهم دیگر این است که هر قسمت برنامه را پس از نوشتن بررسی کنیم و مشکلات برنامه را در همان ابتدا مشخص کنیم. در برنامه‌نویسی یکی از مهمترین موارد عیب‌یابی برنامه است. در اینجا سعی می‌کنیم به اکثر اشتباهات رایج برنامه‌نویسی اشاره‌ای داشته باشیم. مرحله اول برنامه‌ای که می‌خواهیم بنویسیم رسم زمینه بازی است. اما زمینه‌بازی چه چیزی خواهد بود؟ باتوجه به اینکه بازی دوز از یک زمین ۳ در ۳ تشکیل شده‌است می‌توانیم یک آرایه دوبعدی به ابعاد ۳ در ۳ را به عنوان زمین بازی در نظر بگیریم. با توجه به اینکه این آرایه در اکثر توابع ما مورد نیاز است و همچنین در تمام زمان‌ها یک مقدار را خواهد داشت بهتر است آن را به صورت سراسری تعریف کنیم. اما باید توجه داشته باشیم که تا حد امکان باید از تعریف متغیرها به صورت سراسری پرهیز کرد زیرا متغیرهای سراسری عمری برابر عمر برنامه دارند اگر موارد استفاده آنها محدود باشد ممکن است باعث اشغال فضای اضافه شوند. کد برنامه تا به اینجا به شکل زیر است.

```c
#include <stdio.h>
char a[3][3];

int main(){

    return 0;
}
```

نکته دیگری که قبل از شروع کد زدن به آن باید بپردازیم این است که توجه داشته باشید که فاصله در زبان C مگر برای جدا سازی عبارت‌های اصلی، غیر ضروری هستند. پر واقع ما می‌توانیم کد بالا را به شکل

```c
#include <stdio.h>
                char a[3][3];
int


 main()         {
            return 0            ;


                                        }
```

```c
#include <stdio.h> char a[3][3]; int main(){ return 0; }
```

نیز نوشت. اما همانطور که دیده می‌شود کد اول زیباتر و خواناتر است همچنین در صورت بروز مشکل در برنامه ساده تر می‌توان آن را debug کرد. در نتیجه سعی کنید تا جایی که ممکن است کد خود را مرتب بنویسید. همچنین با توجه به اینکه ممکن است که مدت‌ها بعد بخواهیم کد خود را بررسی کنیم پس بهتر است که از comment در کدها استفاده کنیم تا بتوانیم بعد از چند مدت از کد خود سردر بیاریم.

نکته مهم بعدی در مورد اسم متغیرها می‌باشد. در اینجا ما اسم آرایه خود را a انتخاب کردیم درحالی که اسم مناسبی به نظر نمی‌آید. تا جایی که ممکن است موارد زیر را در انتخاب اسم متغیرها و اسم توابع رعایت کنید.

- از تعریف متغیرها با عمر بالا به صورت تک حرفی خودداری کنیم.
- سعی کنیم اسم متغیر را متناسب با کاربرد آن انتخاب کنیم.
- سعی کنید با پسوندها و پیشوندها اطلاعات اضافی به اسمی که انتخاب می‌کنید اضافه کنید.
- از اسم گذاری‌های بسیار طولانی پرهیز کنید.
- از یک ساختار اسم‌گذاری مشخص استفاده کنید برای مثال از underline یا capital letters برای جداسازی کلمات یک اسم استفاده‌کنید(sum_array, sumArray).
- ثابت‌ها را با capital letters مشخص کنید برای مثال (`const int PI = 3.14;`).

پس سعی می‌کنیم که کد خود را به شکل زیر تغییر دهیم.

```
#include <stdio.h>
char board[3][3]; /**   board bazi, az karakter haie 'X' o 'O' baraie
                   *   emtiaz bazi konan va 'B' baraie khane khali
                   *   estefade misavad
                   **/
int main(){

    return 0;
}
```

در ابتدا سعی می‌کنیم متغیرهای خود را مقداردهی کنیم. ابتدا قطعه کدی می‌نویسیم که تمام خانه‌های زمینه ما را با کاراکتر 'B' پر کند. برای اینکار ۲تا حلقه تودرتو کافی است که ۹ خانه مورد نظر را پرکند. قطعه کد را به شکل زیر می‌نویسیم(برای کوتاهی متن از نوشتن کامل commnet در کد خودداری می‌کنیم).

```
#include <stdio.h>
char board[3][3]; /* comment */
int main(){

    for(i = 0 ; i <= 3 ; i++){
        for(i = 0 ; i <= 3 ; i++){
            board[i][j] = 'B';
        }
    }
    return 0;
}
```

پس از اجرای برنامه با خطای زیر رو برو می‌شویم:

/home/karo/... |5|error: 'i' undeclared (first use in this function)|

/home/karo/... |7|error: 'j' undeclared (first use in this function)|

در واقع این خطاها به ما می‌گویند که از متغیرهایی استفاده شده است که تعریف نشده‌اند. اولی i در خط ۱۵ام برنامه و دیگری j در خط ۱۷ام برنامه. حال به تعریف متغیرها اقدام کرده و دوباره برنامه را اجرا می‌کنیم. برنامه تا کنون بدون مشکل اجرا شد. حال سعی کنیم با استفاده از ابزار debug که codeblocks در اختیار ما قرارداده است بررسی کنیم آیا آرایه به طور صحیح مقدار دهی شده است یا خیر.

در codeblocks برروی نوار مابین شماره خط‌ها و کد برنامه کلیک کنید. سپس در قسمت Debug برروی Start / Continue F8 کلیک کنید.

Debug -> Debugging Windows -> Watches

را فعال کرده و در ستون سمت چپ board را تایپ کرده و F7 را میزنیم تا خط به خط برنامه اجرا شود. در پایان می‌بینم که جدول به شکل زیر پر شده است.

B	0	0
B	0	0
B	0	0

اولین مشکل برنامه را پیدا کردیم. همچنین در هنگام اجرا به این نکته توجه داشته‌اید که حلقه داخلی ۴بار اجرا شد و حلقه بیرونی یک بار اجرا شد در حالی که هر دو حلقه باید ۳بار اجرا می‌شدند. با بررسی دوباره کد خواهیم یاقت که هنگامی حلقه را به شکل

```
for(i = 0 ; i <= n ; i++)
```

می‌نویسیم حلقه n+1 بر اجرا می‌شود. پی در واقه باید تساوی‌ها را برداریم. همچنین به اشتباد متغیرهای هر دو حلقه i تعریف شده پس یکی را به j تبدیل می‌کنیم. و کد به شکل زیر خواهد شد.

```
1  #include <stdio.h>
2  char board[3][3]; /* comment */
3  int main(){
4      int i, j;
5      for(i = 0 ; i < 3 ; i++){
6          for(j = 0 ; j < 3 ; j++){
7              board[i][j] = 'B';
8          }
9      }
10     return 0;
11 }
```

اکنون بهتر است که دستورات تابع main را به یک تابع دیگر منتقل کنیم. اینکار باعث می‌شود که فقط دستورات اصلی در تابع main باقی‌بماند در نتیجه خوانایی برنامه بالا رفته و در صورت بروز مشکل میتوانیم بسادگی مشکل برنامه را رفع کنیم. همچنین در صورت نیاز اگر خواستیم که قسمتی دیگر به برنامه اضافه کنیم به سادگی و می‌توانیم اینکار را انجام دهیم و برنامه تغییرات کمی را نیاز دارد. در واقع این نکته را توجه داشته باشید که بهتر است تابع اصلی بازی شما انقدر کوتاه باشد که در یک صفحه قابل نمایش باشد. تغییراتی را که اعمال کردیم در صحفه بعد می‌توانید مشاهده کنید. حال دوباره برنامه را اجرا می‌کنیم که از صحت برنامه اطمینان حاصل کنیم. شاید کار بیهوده‌ای به نظر برسد که بعد از هر تغییر ساده یک بار برنامه را اجرا کنیم اما باید توجه داشته باشیم که یافتن یک مشکل در ۱۰ خط بسیار راحت‌تر از یافتن ۱۰ مشکل در ۱۰۰ خط

است. اگر در ابتدا مشکلات برنامه را حل نکنید برنامه با مشکل جلو رفته و در آخر به مشکلات آن اضافه خواهد شد و در نتیجه debug برنامه برای ما کاری بسیار سخت و گاهی غیر ممکن می‌شود.

```c
#include <stdio.h>
char board[3][3]; /* comment */

int main(){
    install_variable();
    return 0;
}

void install_variable(){
    int i,j;
    for(i = 0 ; i < 3 ; i++){
        for(j = 0 ; j < 3 ; j++){
            board[i][j] = 'B';
        }
    }
}
```

جالب است که بعد از اجرای برنامه با هشدار

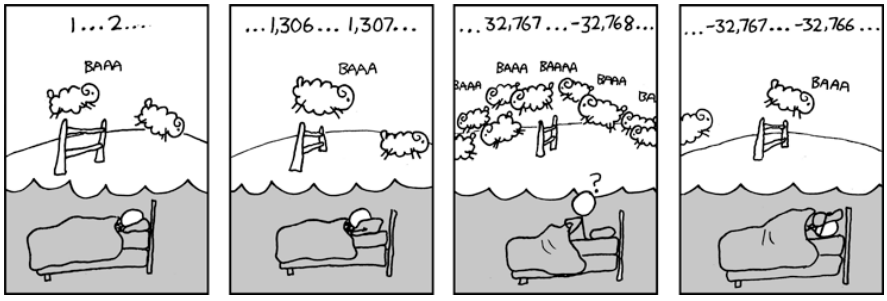
/home/karo/... |5|warning: implicit declaration of function ‘install_variable’ [-Wimplicit-function-declaration]|

/home/karo/... |9|warning: conflicting types for ‘install_variable’ [enabled by default]|

مواجه می‌شویم در واقع در زبان C تعریف نکردن یک تابع قبل از استفاده آن خطا محسوب نمی‌شود اما اگر همین برنامه را با کامپایلر مربوط به زبان C++ که g++ 4.8.2 نام دارد اجرا کنیم پیغام زیر را دریافت خواهیم‌کرد.

main.c:5:22: error: ‘install_variable’ was not declared in this scope install_variable();

در نتیجه بهتر است (در واقع باید) توابعی را قبل از استفاده تعریف کنیم. کد را به شکل زیر تغییر می‌دهیم تا خطای برنامه رفع شود.

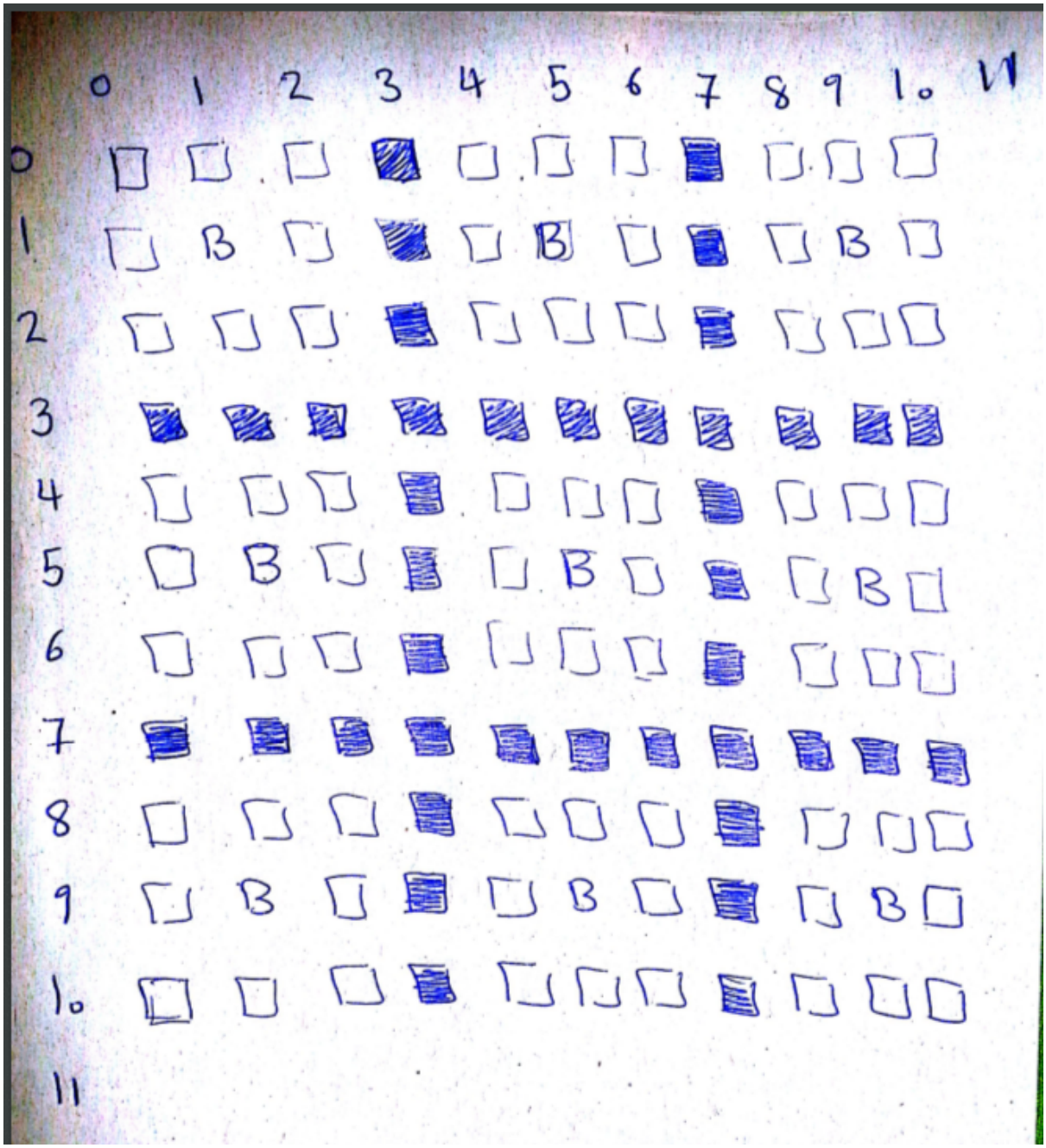


```c
#include <stdio.h>
char board[3][3]; /* comment */

void install_variable(); /* comment */

int main(){
    install_variable();
    return 0;
}

void install_variable(){
    int i,j;
    for(i = 0 ; i < 3 ; i++){
        for(j = 0 ; j < 3 ; j++){
            board[i][j] = 'B';
        }
    }
}
```

از این به بعد برای جلوگیری از افزایش حجم متن از نوشتن بدنه توابعی که در توضیحات به ان نیاز نیست خودداری می‌کنیم.

اکنون با اجرای دوباره برنامه خواهیم دید که خطاهای زبان C++ و هشدارهای زبان C از بین رفته‌اند. اکنون زمان آن فرارسیده‌است که قطعه کدی بنویسیم که برای ما شکل بازی را چاپ کند. در واقع می‌خواهیم که کد شکل زیر را برای ما چاپ کند.

برای اینکار باید محل چاپ کاراکترهارا مشخص کنیم.



همانطور که مشاهده می‌کنید ما برای رسم به دو حلقه تودرتو که هر کدام ۱۱ بار باید اجرا شوند نیاز داریم. فرض کنید متغیری که مشخص کننده ستون است را x و متغیری که مشخص کننده سطر است را y بنامیم.

```
for(y = 0 ; y < 11 ; y++){
    for(x = 0 ; x < 11 ; x++){
        // Code body
    }
}
```

اکنون نکته‌ای که وجود دارد این است که زمانی که باقی‌مانده y به عدد چهار برابر ۳ باشد آنگاه باید یک سطر کامل را کاراکترهای دیواره قرار دهیم در غیر این صورت باید به صورت عادی عمل کنیم. این قسمت را به شکل زیر مشخص می‌کنیم.

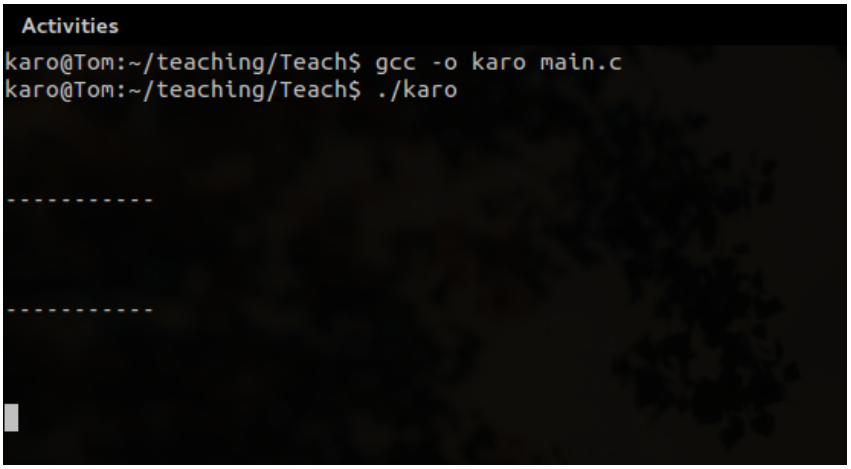
```
for(y = 0 ; y < 11 ; y++){
    for(x = 0 ; x < 11 ; x++){
        if(y % 4 == 3){
            printf("-");
        } else {
            // do the normal
        }
    }
}
```

انتظار داریم که خروجی مشابه شکل باشد با این تفاوت که فقط خط ۳ و ۷ رسم شده باشند و همه جای صفحه سفید باشد اما خروجی ما به شکل زیر است.

دلیل این است که ما بعد از پیمایش هر سطر به خط بعدی نرفته‌ایم. پس باید کد را به شکل زیر تغییر دهیم

```
for(y = 0 ; y < 11 ; y++){
    for(x = 0 ; x < 11 ; x++){
        if(y % 4 == 3){
            printf("-");
        } else {
            // do the normal
        }
    } printf("\n");
}
```

بعد از اجرای دوباره برنامه خروجی زیر را مشاهده می‌کنیم.



و اما در مورد سطرهای دیگر ۳ حالت به وجود می آید

- اگر باقی‌مانده x به عدد چهار برابر سه باشد آنگاه کاراکتر دیواره باید چاپ شود.
- اگر باقی‌مانده x به چهار برابر یک و همچنین باقی مانده y نیز برچهار برابر یک باشد. آنگاه باید خانه board[x-1/4][y-1/4] چاپ شود.
- در غیر این صورت باید کاراکتر فضای خالی چاپ شود.

برنامه آن به شکل زیر است.

```
1  for(y = 0 ; y < 11 ; y++){
2      for(x = 0 ; x < 11 ; x++){
3          if(y % 4 == 3){
4              printf("-");
5          } else {
6              if(x % 4 == 3){
7                  printf("|");
8              } else if(x % 4 == 1 && y % 4 == 1){
9                  printf("%c",board[x-1/4][y-1/4]);
10             } else {
11                 printf(" ");
12             }
13         }
14     } printf("\n");
15 }
```

پس از اجرای برنامه خروجی زیر را دریافت خواهیم کرد. اما خروجی صحیح نیست. تعداد فضاهای خالی و همچنین تعداد دیواره صحیح چاپ شده است اما تعداد کاراکترهای board صحیح نیست پس به نظر می آید در خط ۹ برنامه مشکلی وجود داشته باشد.فرض می‌کنیم که x = 9, y = 5 باشد.

در این صورت انتظار داریم که board[2][1] چاپ شود. حال مقادیر را جایگذاری می‌کنیم

$$\texttt{board[x-1/4][y-1/4]} \rightarrow \texttt{board[9-1/4][5-1/4]} \rightarrow \texttt{board[9-0][5-0]} \rightarrow \texttt{board[9][5]}$$

مشاهده می‌کنیم که ما به اولویت عملگرها توجهی نداشته‌ایم در واقع اولویت به تقسیم است و ابتدا حاصل 1/4 محاسبه می‌شود و جون هر دو آن‌ها اعدادی از نوع int هستند حاصل آن‌ها نیز از نوع int بوده و درنتیجه مقدار صفر را به ما برمی‌گردانند. با اضافه کردن پرانتز اولویت‌ها زا تغییر داده و کد را تصیح می‌کنیم همچنین آن را به یک تابع منتقل می‌کنیم. همچنین پس از اطمینان از صحت برنامه کد را طوری تغییر می‌دهیم که اگر board[(x-1)/4][(y-1)/4] حاوی مقدار B بود آنگاه کاراکتر فضای خالی چاپ کند.

```
1  void print_board(){
2      int x, y;
3      for(y = 0 ; y < 11 ; y++){
4          for(x = 0 ; x < 11 ; x++){
5              if(y % 4 == 3){
6                  printf("-");
7              } else {
8                  if(x % 4 == 3){
9                      printf("|");
10                 } else if(x % 4 == 1 && y % 4 == 1){
11                     if(board[(x-1)/4][(y-1)/4] == 'B') printf(" ");
12                     else printf("%c",board[(x-1)/4][(y-1)/4]);
13                 } else {
14                     printf(" ");
15                 }
16             }
17         } printf("\n");
18     }
19 }
```

اکنون کافی است که روال بازی را پیاده‌سازی کنیم. بازی دوز یک بازی نوبتی است در نتیجه:

(۱) ما باید در هر لحظه از بازی باخبر باشیم که نوبت کدام یک از بازیکن هاست که بازی کنند.

همچنین باید توجه داشته باشیم که حرکت غیرمجاز انجام نشود. حرکت‌های غیر مجاز را به شکل زیر تعریف می‌کنیم

- نباید در خانه‌ای که قبلا مهره گذاشته شده مهره جایگذاری شود.
- نباید به خانه از خارج از زمینه بازی برای حرکت اشاره شود.
- نباید کاراکترهای غیرمجاز به عنوان آدرس تحلیل و بررسی شوند. برای مثال هنگامی اکه از ما می‌پرسد که می‌خواهید در کدام خانه مهره خود را قرار دهید ما به جای مختصات صحیح عددی اعشاری را وارد کنیم یا با حروف مشخص کنیم که در کدام خانه می‌خواهیم مهره خود را قرار دهیم.

بررسی دو مورد اول ساده است و در طول پیاده‌سازی برنامه بررسی‌های لازیم را انجام می‌دهیم اما در سومین مورد کار کمی پیچیده‌تر است و حل آن را به انتهای مقاله موکول می‌کنیم.

مورد بعدی بررسی پایان بازی است. در واقع بازی در دو حالت پایان خواهد یافت

- یکی از بازیکن ها برنده بازی شده باشد.
- همه مهره‌ها در زمینه بازی قرار داده شده باشد.

باید توجه داشته باشیم که

(۲) بعد از هر حرکت باید بررسی کنیم که آیا بازیکنی که حرکت را انجام داده است بازی را پیروز شده است یا خیر.

یک متغیر از نوع char تعریف می‌کنیم و نام آن را player می‌گذاریم این متغیر زمانی که مقدار X را دارد نوبت بازیکنی است که X را در خانه‌ها قرار می‌دهد و زمانی که مقدار O را دارد نوبت بازیکن دیگر است. در ابتدا فرض می‌کنیم که نوبت بازیکن X باشد که بازی را آغاز کند. در این صورت در تابع install_variable() باید متغیر player مقداردهی شود.کد برنامه به شکل زیر خواهد بود.

```
1  #include <stdio.h>
2  char board[3][3];           /* game board,
3                               * X and O for player move
4                               * B for Blank */
5  char player;                /* X when first player plays, otherwise O */
6  int game_ends;              /* true(1) if game ends, otherwise false(0) */
7
```

```
8  /* * * * * * * * Functions * * * * * * * * */
9  void install_variable();     /* comment */
10 void print_board();          /* comment */
11 int player_move();           /* comment */
12 int wins_game();             /* comment */
13 void end_of_game(char);      /* comment */
14 int no_more_move();          /* comment */
15
16 /* * * * * * * * main() * * * * * * * * */
17 int main(){
18     install_variable();
19     print_board();
20
21     /* start game */
22     while(!game_ends){
23         player_move();
24
25         if(wins_game(player))
26             end_of_game(player);
27         else if(no_more_move())
28             end_of_game('B');
29     }
30
31     return 0;
32 }
33
34 /* * * * * * * * player_move() * * * * * * * * */
35 int player_move(){
36     return 0;
37 }
38
39 /* * * * * * * * wins_game() * * * * * * * * */
40 int wins_game(){
41     return 0;
42 }
43
44 /* * * * * * * * end_of_game(char) * * * * * * * * */
45 void end_of_game(char c){ }
46
47 /* * * * * * * * no_more_move() * * * * * * * * */
48 int no_more_move(){
49     return 0;
50 }
```

```
51
52 /* * * * * * * * install_variable() * * * * * * * * */
53 void install_variable(){
54     /* install board */
55     int i,j;
56     for(i = 0 ; i < 3 ; i++){
57         for(j = 0 ; j < 3 ; j++){
58             board[i][j] = 'B';
59         }
60     }
61     /* install variables */
62     player = 'X';
63     game_ends = 0;
64 }
65
66 /* * * * * * * * print_board() * * * * * * * * */
67 void print_board(){
68     int x, y;
69     for(y = 0 ; y < 11 ; y++){
70         for(x = 0 ; x < 11 ; x++){
71             if(y % 4 == 3){
72                 printf("-");
73             } else {
74                 if(x % 4 == 3){
75                     printf("|");
76                 } else if(x % 4 == 1 && y % 4 == 1){
77                     if(board[(x-1)/4][(y-1)/4] == 'B') printf(" ");
78                     else printf("%c",board[(x-1)/4][(y-1)/4]);
79                 } else {
80                     printf(" ");
81                 }
82             }
83         } printf("\n");
84     }
85 }
```

تا به اینجای کار برنامه مشکلی ندارد. در اینجا بدنه اصلی برنامه را ساختیم و کم کم توابع را به ترتیب کامل می‌کنیم. توجه داشته باشید که توابع را به ترتیب اجرا کامل کنید که فرصت debug را داشته باشیم و هر مرحله بتوانیم مشکلات برنامه را بیابیم. از تابع `player_move()` آغاز می‌کنیم. در این تابع باید از ما بپرسد که کدام حرکت را می‌خواهیم انجام دهیم. اگر حرکت ما مجاز بود آنگاه اعمال می‌کنیم و مقدار یک را بر می‌گردانیم در غیر این صورت حرکتی انجام نمی‌دهیم و مقدار صفر را برخواهیم گرداند. کد مربوط به این

تابع به شکل زیر خواهد بود.

```
int player_move(){
    int x, y;
    printf("\n* * * Player '%c' turn * * *\n",player);
    printf("Player '%c' please enter two numbers, x & y [0,1,2]: ",player);
    scanf("%d%d",x,y);
    if(board[x][y] == 'B'){
        board[x][y] = player;
        return 1;
    } else {
        return 0;
    }
}
```

ابتدا از کاربر ۲ عدد را گرفته و سپس اگر در خانه مربوطه علامتی قرار نگرفته بود کاراکتر مربوط به بازیکنی که حرکت را انجام داده است را در خانه مربوط قرار می‌دهد. بعد از اجرای برنامه پس از وارد کردن دو عدد برنامه از بین می‌رود. ظاهرا مشکلی در کد دیده می‌شود. اگر دقت کنیم در خط ۵ هنگامی که دو عدد را از ورودی خوانیم به جای آدرس متغیرها خود آن‌ها را به تابع scanf() فرستادیم. بعد از تغیر خط ۵ به شکل

```
scanf("%d%d",&x,&y);
```

برنامه را دوباره از نو اجرا می‌کنیم. اینبار اعداد را درست از ورودی دریافت کرد اما دوباره صفحه بازی را چاپ نکرد. باید در تابع main در حلقه اصلی بازی بعد از هر حرکت زمینه را دوباره از اول چاپ کنیم. همچنین باید یادمان باشد که هر بار بازیکن را تغییر دهیم لذا تابعی می‌نویسیم که نوبت را تغییر دهد.

```
while(!game_ends){
    player_move();
    print_board();
    if(wins_game(player))
        end_of_game(player);
    else if(no_more_move())
        end_of_game('B');
    change_player();
}
```

```
void change_player() { if(player == 'X') player = 'O'; else player = 'X'; }
```

برنامه را دوباره اجرا می‌کنیم و سعی می‌کنیم که خانه تکراری را وارد کنیم. اتفاقی که می‌افتد این است که خانه تغییری نمی‌کند اما نوبت عوض می‌شود. دلیل آن است که خط ۲ برنامه فقط یک بار اجازه حرکت را به یک بازیکن می‌دهد. باید تا زمانی که حرکت صحیح را انجام می‌دهد اجازه حرکت به او داده شود. پس قطعه کد را با تغییری ساده به شکل زیر در می‌آوریم

```
1  while(!game_ends){
2      while(player_move()){
3          printf("Illegal move: Try Again");
4      }
5      print_board();
6      if(wins_game(player))
7          end_of_game(player);
8      else if(no_more_move())
9          end_of_game('B');
10     change_player();
11 }
```

همچنین شرط دیگری به تابع palyer_move() اضافه می‌کنیم که اگر اعداد وارد شده خارج از محدوده بازی باشد اجازه حرکت داده نشود. و تابع نهایی به شکل زیر خواهد بود.

```
1  int player_move(){
2      int x, y;
3      printf("\n* * * Player '%c' turn * * *\n",player);
4      printf("Player '%c' please enter two numbers, x & y [0,1,2]: ",player);
5      scanf("%d%d",&x,&y);
6
7      if(x < 0 || x > 3 || y < 0 || y > 3)
8          return 0;
9
10     if(board[x][y] == 'B'){
11         board[x][y] = player;
12         return 1;
13     } else {
14         return 0;
15     }
16 }
```

اکنون نوبت به تابع wins_game(char) رسیده که تشخیص دهد آیا بازیکنی که حرکت انجام داده بازی را برنده شده‌است یا خیر. فقط کافی است که چند جهتی که ممکن است برد در آن صورت گرفته باشد را

بررسی کنیم. خانه‌های هر جهت در زیر مشخص شده‌اند.

- Vertical
  - (0,0) - (0,1) - (0,2)
  - (1,0) - (1,1) - (1,2)
  - (2,0) - (2,1) - (2,2)
- horizontal
  - (1,0) - (1,0) - (2,0)
  - (1,1) - (1,1) - (2,1)
  - (1,2) - (1,2) - (2,2)
- diagonal
  - (0,0) - (1,1) - (2,2)
  - (0,2) - (1,1) - (2,0)

با استفاده از قطعه کد زیر به سادگی می‌توانیم به بررسی هر سه مورد بپردازیم.

```
1  int wins_game(){
2      int i;
3      for(i = 0 ; i < 3 ; i++){
4          if(board[0][i] + board[1][i] + board[2][i] == 3 * player){
5              return 1;
6          } else if(board[i][0] + board[i][1] + board[i][2] == 3 * player){
7              return 1;
8          }
9      }
10
11     if(board[0][0] + board[1][1] + board[2][2] == 3 * player){
12         return 1;
13     } else if(board[0][2] + board[2][0] + board[1][1] == 3 * player){
14         return 1;
15     }
16
17     return 0;
18 }
```

و در نهایت دو تابع باقی‌مانده را به شکل زیر پیاده می‌کنیم

```
void end_of_game(char c){
    switch(c){
    case 'X':
        printf("Player X is Winner of the Game\n");
        break;
    case 'O':
        printf("Player O is Winner of the Game\n");
        break;
    case 'B':
        printf("Game has no Winner\n");
        break;
    }
    game_ends = 1;
}

int no_more_move(){
    int i, j;
    for(i = 0 ; i < 3 ; i++){
        for(j = 0 ; j < 3 ; j++){
            if(board[i][j] == 'B'){
                return 0;
            }
        }
    }
    return 1;
}
```

در تابع `win_game()` ما از کد ASCII برای مقایسه کاراکترها استفاده کردیم. درواقع می‌دانیم که کاراکتر X برای مثال کد 71 را دارد. آنگاه اگر هر سه خانه‌ی `board[0][0], board[1][1], board[2][2]` مقدار X را در خود ذخیره کرده باشند آنگاه مجموع آن‌ها برابر $71 + 71 + 71$ خواهد بود و در صورتی که با ۳برابر مقدار متغییر `player` برابر باشد آنگاه می‌توانیم نتیجه بگیریم که

```
board[0][0] == board[1][1] == board[2][2] == 'X'
```

امااین استدلال همیشه صحیح نیست زیرا ممکن است مقادیر خانه‌های board متفاوت بوده ولی باز همان عدد را تولید کنند مانند $72 + 71 + 70$ اما در اینجا این اتفاق رخ نخواهد داد زیرا که ما فقط ۳ مقدار در خانه‌های board دخیره کرده‌ایم و این مقادیر تصاعد حسابی تشکیل نمی‌دهند و در نتیجه می‌توان از این روش برای ساده سازی محاسبات استفاده کنیم.

در نهایت می‌توانید برنامه را اجرا کنید و سعی کنید مشکلاتی را در کد بیابید و آن‌هارا حل کنید. برای مثال یکی از مشکلات کد این است که اگر به جای عدد در قسمت ورودی رشته وارد کنیم برنامه بهم خواهد ریخت. می‌توانید به عنوان تمرین به جای عدد رشته از ورودی گرفته و بررسی کنید که آیا رشته ورودی شامل ۱ عدد است یا خیر و اگر یک عدد بود آیا عدد وارد شده در بازه مورد نظر ما قرار دارد یا نه. یکی دیگر از مشکلات برنامه این است که ممکن است کاربر به جای ۲ عدد ۴ عدد را وارد کند آنگاه در واقع همزمان ۲ حرکت با هم انجام می‌شود. در واقع وقتی شما کلیدی را وارد می‌کنید کلید وارد یک صف در سیستم عامل می‌شود و هر گاه برنامه شما به `scanf()` یا هر تابع دیگری که وظیفه خواندن ورودی را از کاربر دارد برسد یکی از اجزای صف را برمی‌دارد. همچنین تابعی به نام `kbhit()` در کتابخانه `conio.h` وجود دارد که در صورتی که این صف خالی باشد صفر و در غیر این صورت ۱ را برمی‌گرداند. همچنین تابع `getch()` یک کاراکتر از این صف را بدن اینکه نمایش دهد از صف خارج می‌کند. می‌توانید با استفاده از حلقه‌ها و این توابع صف مورد نظر را خالی کنید. همچنین می‌توانید به عنوان تمرین قسمت‌هایی دیگر به برنامه بیفزایید برای مثال می‌توانید این امکان را در برنامه قرار دهید که کاربر بعد از پایان بازی بدون خروج از بازی یازی جدیدی را شروع کند. در قسمت پایانی می‌توانید کد کامل برنامه را یک جا مشاهده کنید.

در پایان سعی کنید لیستی از خطاهایی که در برنامه‌نویسی دچار آن شده‌اید تهیه کنید و هر بار که برنامه شما اجرا نشد و با مشکل روبرو شدید می‌توانید لیست را بررسی کنید که ایا اشتباهی را دوباره مرتکب شده‌اید. در غیر این صورت پس یافت مشکل آن را به لیست اضافه کنید. برای مثال لیستی از اشتباهات رایج را می‌توانید مشاهده کنید.

- تعریف نکردن متغییرها

```c
int main(){
    scanf("%d",&n);
}
```

- کم بودن طول آرایه‌ها و رشته‌ها

```c
int main(){
    char my_name[4];
    scanf("%s",my_name);
    /* input is "karosahafi" */

    printf("%s",my_name);
    /* output is "karosahafiAborted (core dumped)" */
}
```

- مقداردهی نکردن متغییرها

```c
int main(){
    double var;
    printf("%lf\n",var);
}
// output
// 565846.41552
```

- استفاده از = به جای == در شرط‌ها

```c
int main(){
    int num = 10;
    if(num = 11){
        // In ghesmat az code ejra khahad shod
    }
}
```

- اتمام حلقه قبل از اجرای دستورات آن

```c
int main(){
    int sum = 0, i;
    for(i = 1 ; i <= 100 ; i++);
        sum = sum + i;

    printf("%d\n",sum); // output will be 101 instead of 5050
}
```

- خروج از محدوده آرایه

```c
int main(){
    int arr[10]
    for(i = 1 ; i <= 10 ; i++)
        printf("%d",arr[i]);
}
```

- تقسیم دو عدد صحیح برهم

```c
int main(){
    int a = 7, b = 2;
    double c = a/b;
    printf("%f\n",c);        // output: 3.00000
    c = (double)a/b;
    printf("%f\n",c);        // output: 3.50000
}
```

- چند بار تعریف یک متغیر

```c
int main(){
    int i, j;
    for(i = 0 ; i < 10 ; i++) j = i;
    int i = 10;
    printf("%d\n",i);
}
```

- فراموش کردن `break` در `switch`

```c
int main(){
    int x = 2;
    switch(x) {
    case 2:
      printf("Two\n");
    case 3:
      printf("Three\n");
    }
}
```

- فراموش کردن `&` در `scanf`

```c
int main(){ int x; scanf("%d",x); }
```

```c
/* in the name of God - tic tac toe project */

#include <stdio.h>
char board[3][3];           /* game board,
                             * X and O for player move
                             * B for Blank */
char player;                /* X when first player plays, otherwise O */
int game_ends;              /* true(1) if game ends, otherwise false(0) */

void install_variable();    /* comment */
void print_board();         /* comment */
int player_move();          /* comment */
int wins_game();            /* comment */
void end_of_game(char);     /* comment */
int no_more_move();         /* comment */
void change_player();       /* comment */

int main(){
    install_variable();
    print_board();

    /* start game */
    while(!game_ends){
        while(!player_move()){
            printf("Illegal move: Try Again");
        }
        print_board();
        if(wins_game())
            end_of_game(player);
        else if(no_more_move())
            end_of_game('B');
        change_player();
    }

    return 0;
}

void change_player() {
    if(player == 'X') player = 'O';
    else player = 'X';
}

```

```
43  int player_move(){
44      int x, y;
45      printf("\n* * * Player '%c' turn * * *\n",player);
46      printf("Player '%c' please enter two numbers, x & y [0,1,2]: ",player);
47
48      scanf("%d%d",&x,&y);
49
50      /* comment */
51      if(x < 0 || x > 3 || y < 0 || y > 3){
52          return 0;
53      }
54
55      /* comment */
56      if(board[x][y] == 'B'){
57          board[x][y] = player;
58          return 1;
59      } else {
60          return 0;
61      }
62  }
63
64  int wins_game(){
65      int i;
66
67      /* comment */
68      for(i = 0 ; i < 3 ; i++){
69          if(board[0][i] + board[1][i] + board[2][i] == 3 * player){
70              return 1;
71          } else if(board[i][0] + board[i][1] + board[i][2] == 3 * player){
72              return 1;
73          }
74      }
75
76      /* comment */
77      if(board[0][0] + board[1][1] + board[2][2] == 3 * player){
78          return 1;
79      } else if(board[0][2] + board[2][0] + board[1][1] == 3 * player){
80          return 1;
81      }
82
83      return 0;
84  }
85
```

```
86  void end_of_game(char c){
87      switch(c){
88      case 'X':
89          printf("Player X is Winner of the Game\n");
90          break;
91      case 'O':
92          printf("Player O is Winner of the Game\n");
93          break;
94      case 'B':
95          printf("Game has no Winner\n");
96          break;
97      }
98      game_ends = 1;
99  }
100
101 int no_more_move(){
102     int i, j;
103     for(i = 0 ; i < 3 ; i++){
104         for(j = 0 ; j < 3 ; j++){
105             if(board[i][j] == 'B'){
106                 return 0;
107             }
108         }
109     }
110     return 1;
111 }
112
113 void install_variable(){
114     /* install board */
115     int i,j;
116     for(i = 0 ; i < 3 ; i++){
117         for(j = 0 ; j < 3 ; j++){
118             board[i][j] = 'B';
119         }
120     }
121     /* install variables */
122     player = 'X';
123     game_ends = 0;
124 }
125
126
127
128
```

```
129 void print_board(){
130     int x, y;
131     for(y = 0 ; y < 11 ; y++){
132         for(x = 0 ; x < 11 ; x++){
133             if(y % 4 == 3){
134                 printf("-");
135             } else {
136                 if(x % 4 == 3){
137                     printf("|");
138                 } else if(x % 4 == 1 && y % 4 == 1){
139                     /* comment */
140                     if(board[(x-1)/4][(y-1)/4] == 'B') printf(" ");
141                     /* comment */
142                     else printf("%c",board[(x-1)/4][(y-1)/4]);
143                 } else {
144                     printf(" ");
145                 }
146             }
147         } printf("\n");
148     }
149 }
```