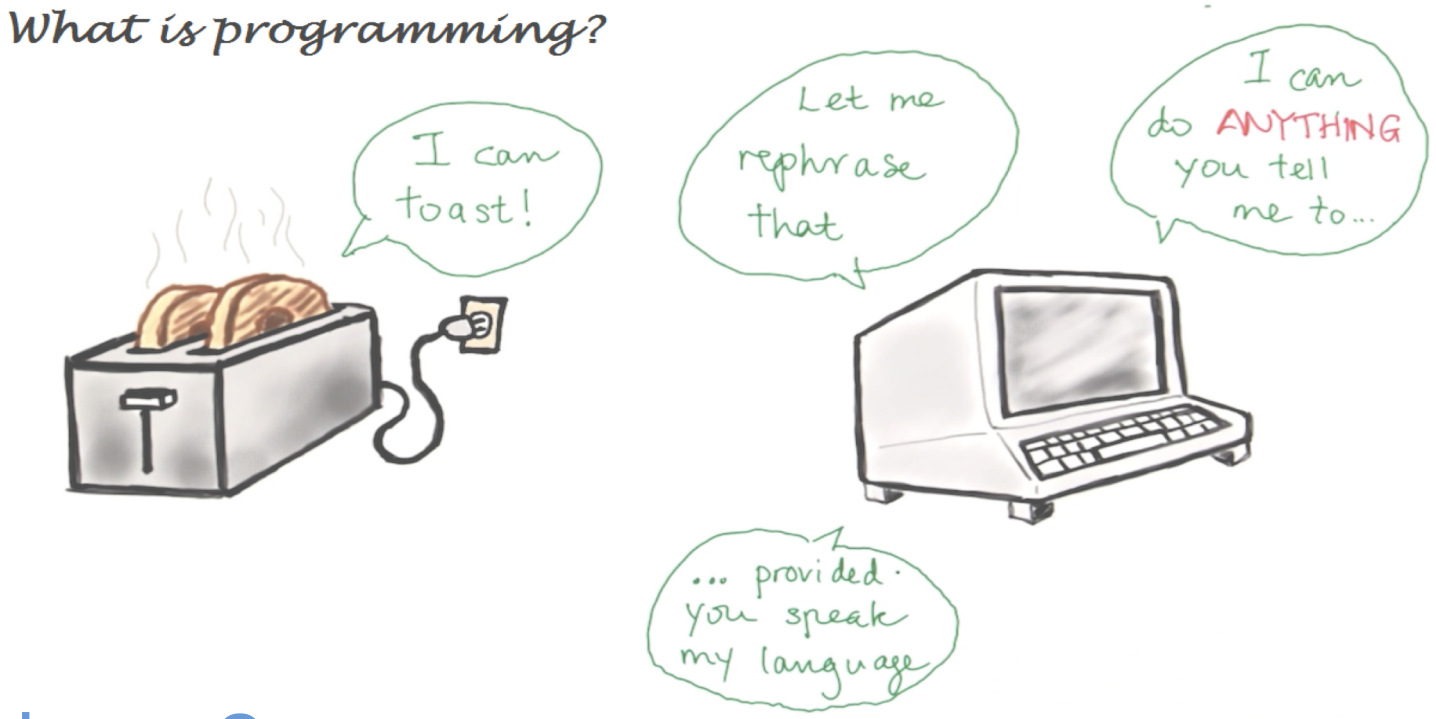


Nội dung 8

Bài ôn tập số 1

CT101 – Lập trình căn bản

Khoa CNTT&TT – ĐHCT

Mục đích

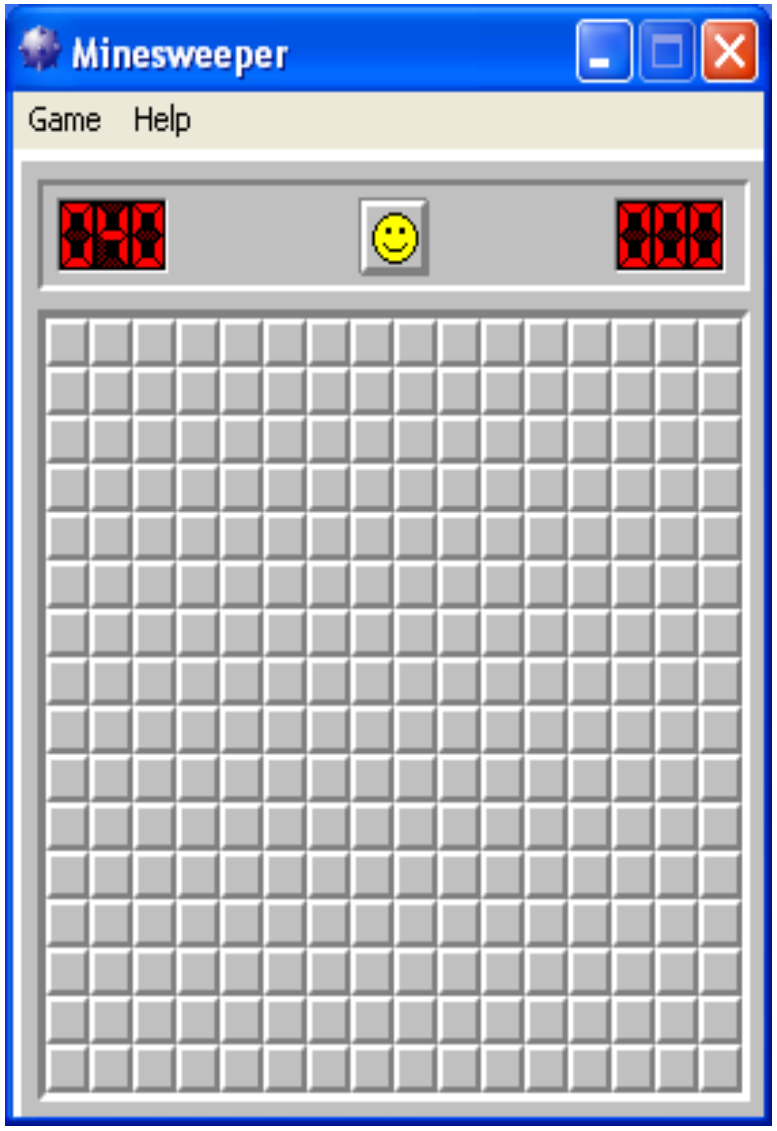
- Củng cố các nội dung đã học:
  - Mảng
  - Chương trình con
  - Truyền tham số
  - Đệ quy
- Vận dụng kiến thức đã học vào việc thiết kế và cài đặt một phần mềm đơn giản (trò chơi)
- Rèn luyện kỹ năng thực hành

Nội dung

- Giới thiệu về trò chơi **gỡ mìn** (Minesweeper)
- Phân tích trò chơi
- Thiết kế trò chơi **SimpleMinesweeper**
  - Xây dựng lưu đồ tổng quát của chương trình
  - Biểu diễn dữ liệu
  - Xác định các chương trình con, lập trình
  - Lắp ráp, chạy và kiểm thử từng phần
    - Bài tập 1
    - Bài tập 2
- Nâng cao

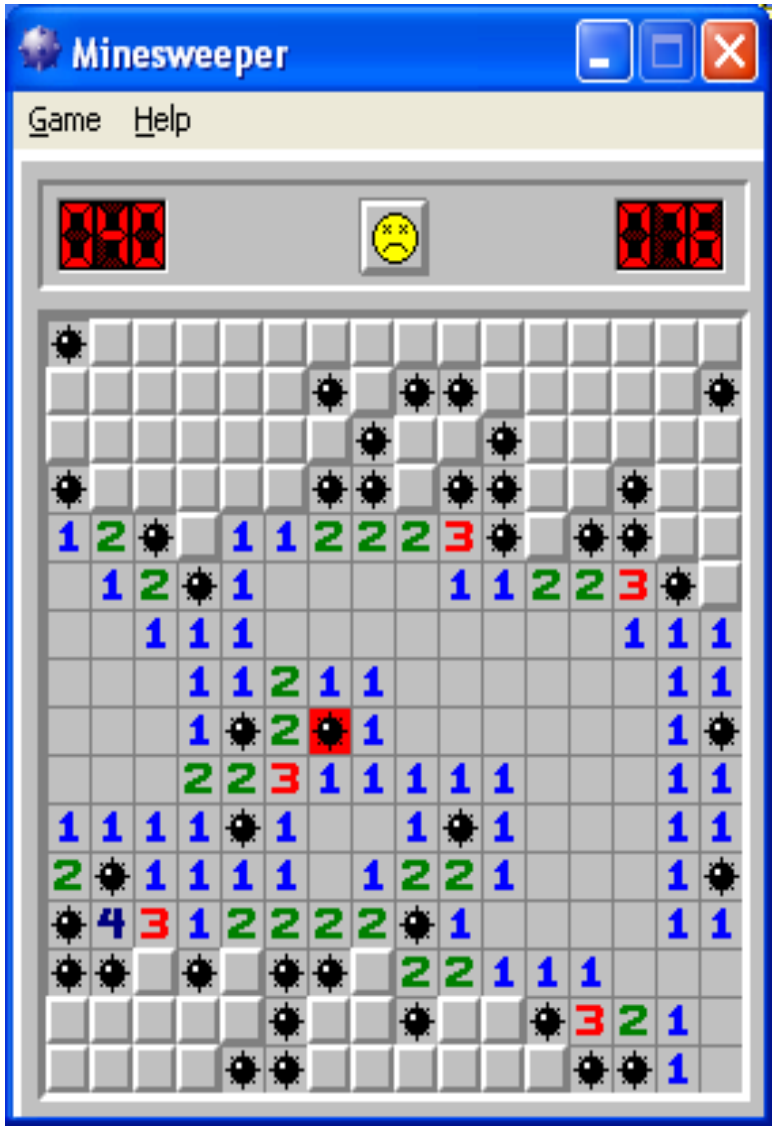
Giới thiệu về trò chơi gỡ mìn

- Trò chơi kinh điển của Microsoft đi kèm với hệ điều hành Windows
- Tác giả: Robert Donner và Curt Johnson



Phân tích trò chơi gỡ mìn

- Bản đồ mìn:
  - Bảng kích thước **m x n**
  - Có **K** quả mìn nằm ngẫu nhiên ở **K** ô
- Luật chơi:
  - Người dùng chuột chọn 1 ô để mở
  - Nếu ô được chọn có mìn => mìn nổ, trò chơi kết thúc
  - Nếu ô được chọn không có mìn nó sẽ được mở ra theo **"Quy tắc mở ô lân cận"**.
  - Trò chơi kết thúc khi tất cả các ô không có mìn đều được mở ra.



Phân tích trò chơi gỡ mìn

- Quy tắc mở ô lân cận:
  - Khi một ô được mở ra, ta sẽ nhìn thấy có một con số nằm bên dưới nó. Con số này mô tả số quả mìn nằm trong 8 ô xung quanh nó.
  - Nếu xung quanh ô mở ra không có quả mìn nào (ô trống) => 8 ô xung quanh nó sẽ tự động được mở ra.
  - Nếu ô được mở ra không phải ô trống => bạn có dám mở các ô lân cận ra không ?
  - Tiếp tục xét các ô vừa mới mở ra theo quy tắc trên.

SimpleMinesweeper

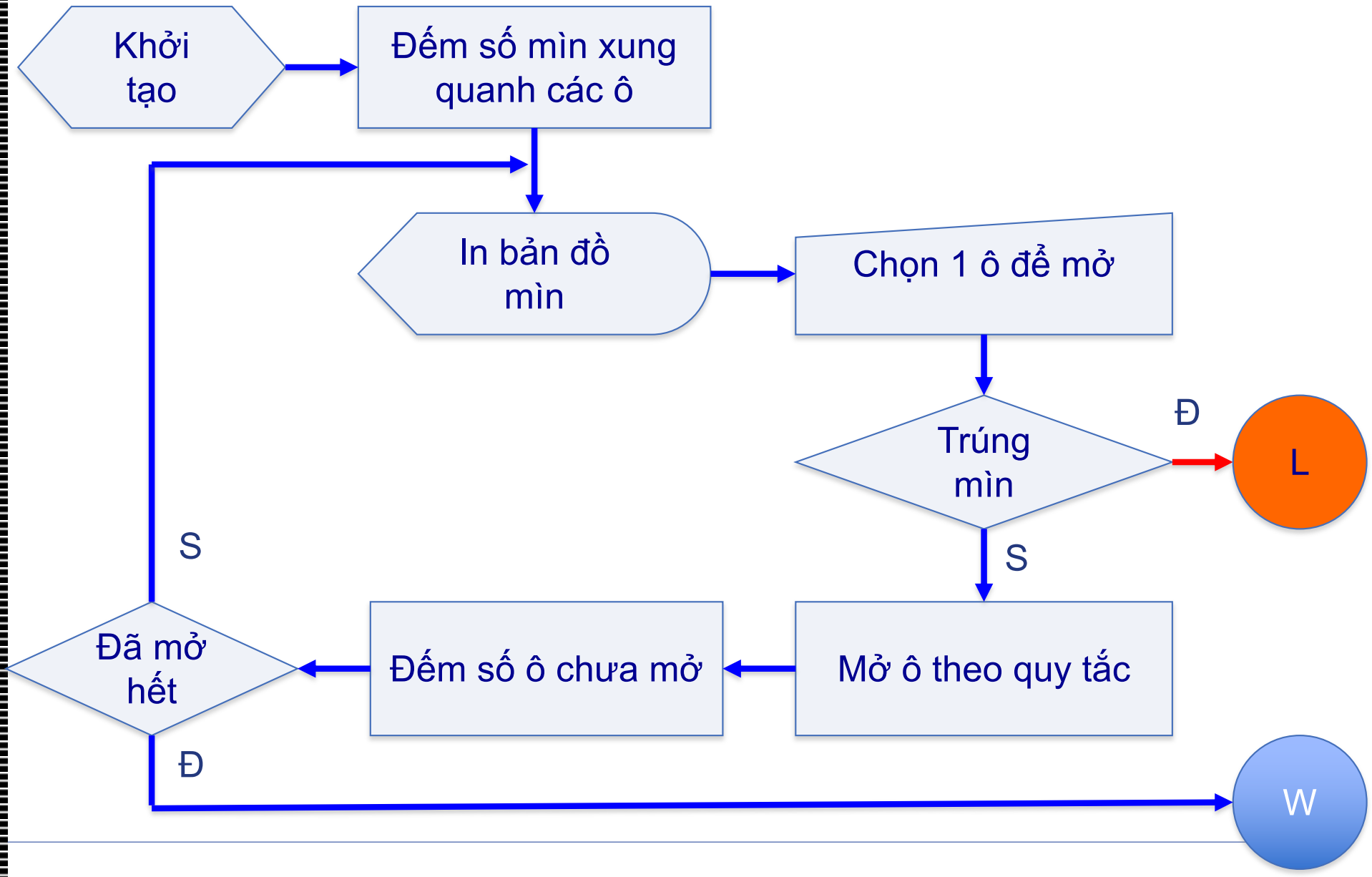
- Minesweeper:
  - Dựa trên trò chơi Minesweeper
  - Tính năng giống hệt Minesweeper
- Simple:
  - Không sử dụng chuột => chọn ô để mở bằng cách chỉ định vị trí của nó
  - Không sử dụng giao diện đồ hoạ => nhập dữ liệu bằng bàn phím, sử dụng các ký tự đặc biệt để hiển thị bản đồ mìn

SimpleMinesweeper

```
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
 0 # # # # # # # # # # # # # # # #
 1 # # # # # # # # # # # # # # # #
 2 # # # # # # # # # # # # # # # #
 3 # # # # # # # # # # # # # # # #
 4 # # # # # # # # # # # # # # # #
 5 # # # # # # # # # # # # # # # #
 6 # # # # # # # # # # # # # # # #
 7 # # # # # # # # # # # # # # # #
 8 # # # # # # # # # # # # # # # #
 9 # # # # # # # # # # # # # # # #
10 # # # # # # # # # # # # # # # #
11 # # # # # # # # # # # # # # # #
12 # # # # # # # # # # # # # # # #
13 # # # # # # # # # # # # # # # #
14 # # # # # # # # # # # # # # # #
15 # # # # # # # # # # # # # # # #
Nhập ô cần mở:
```

```
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
 0 # # # # # # # # # # # # # # # #
 1 # # # # # # # # # # # # # # # #
 2 # # # # # # # # # # # # # # # #
 3 # # # # # # # # # # # # # # # #
 4 # # # # # # # # 2 # # # # # # #
 5 # 2 1 1 # # # # # # # # # # # #
 6 # 1 . 1 # # # # # # # # # # # #
 7 # 1 . 2 # # # # # # # # # # # #
 8 # 1 2 4 # # # # # # # # # # # #
 9 # # # # # # # # # # # # # # # #
10 # # # # # # # # # # # # # # # #
11 # # # # # # # # # # # # # # # #
12 # # # # # # # # # # # # # # # #
13 # # # # # # # # # # # # # # # #
14 # # # # # # # # # # # # # # # #
15 # # # # # # # # # # # # # # # #
```

Lưu đồ tổng quát

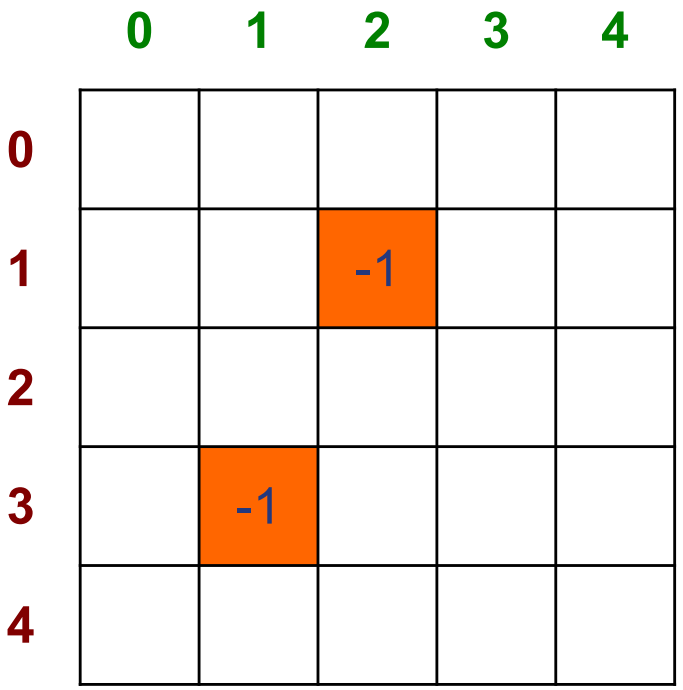


Biểu diễn dữ liệu

- Bản đồ mìn:
  - Sử dụng mảng 2 chiều để lưu bản đồ mìn
  - Mỗi phần tử của mảng lưu trữ số mìn xung quanh ô tương ứng
- Trạng thái của các ô:
  - Mảng 2 chiều lưu trạng thái của các ô, ví dụ: 0 (chưa mở), 1 (đã mở)

Khai báo

```
#include <stdio.h>
#include <stdlib.h>

#define M 5
#define N 5
#define MINE -1

/*Mảng B lưu bảng đồ mìn*/
int B[M][N];

/*Mảng T lưu trạng thái các ô*/
int T[M][N];
```



- Bảng đồ mìn có 5 hàng x 5 cột
- Quy ước: ô chứa mìn có giá trị -1
- Có 2 quả mìn nằm ở vị trí (1, 2) và (3, 1)
- **Yêu cầu: viết hàm init() khởi tạo bản đồ mìn như hình vẽ**

Khởi tạo

```
void init() {
  for (int i = 0; i < M; i++)
    for (int j = 0; j < N; j++) {
      B[i][j] = 0;
      T[i][j] = 0;
    }
    B[1][2] = MINE;
    B[3][1] = MINE;
}
```

	0	1	2	3	4
0	0	0	0	0	0
1	0	0	-1	0	0
2	0	0	0	0	0
3	0	-1	0	0	0
4	0	0	0	0	0

- Khởi tạo mảng B chứa toàn số 0 (2 vòng for), mảng T cũng chứa toàn số 0 (chưa ô nào được mở hết)
- Ô (1, 2) và (3, 1) có giá trị -1 ứng với vị trí 2 quả mìn
- **Yêu cầu: hãy tính số mìn xung quanh của tất cả các ô còn lại (tính thủ công)**

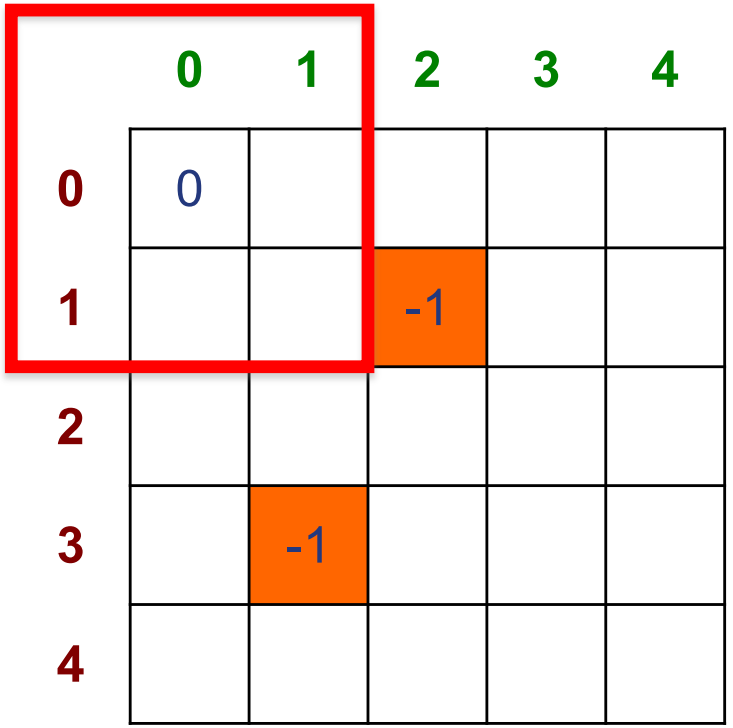
Đếm số mìn

- Bảng đồ mìn có 5 hàng, 5 cột
- Có 2 quả mìn nằm ở vị trí (1, 2) và (3, 1)
- **Yêu cầu: hãy viết hàm count_mines() để tự động đếm số mìn xung quanh của tất cả các ô trong bản đồ mìn (mảng B)**

	0	1	2	3	4
0	0	1	1	1	0
1	0	1	-1	1	0
2	1	2	2	1	0
3	1	-1	1	0	0
4	1	1	1	0	0

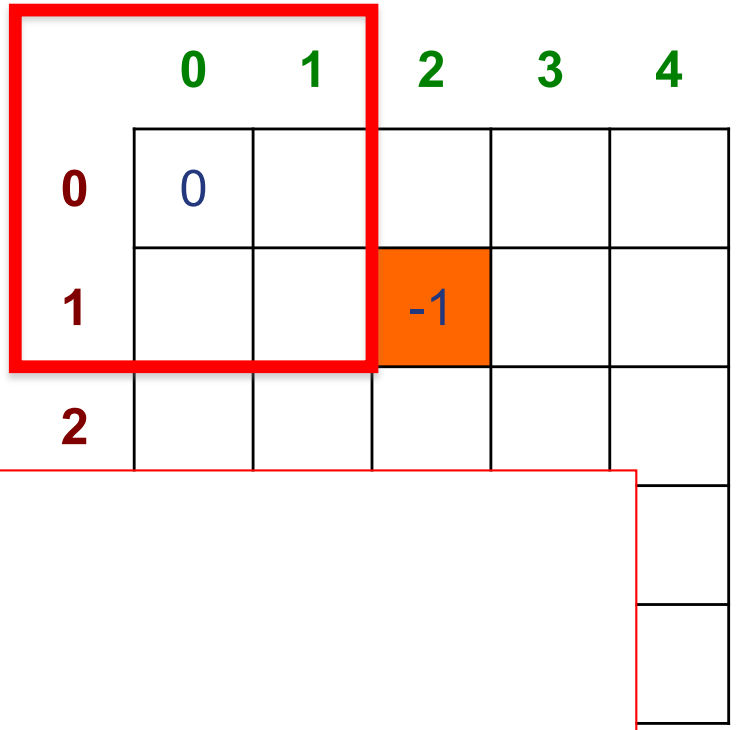
Đếm số mìn

- Duyệt qua từng ô của mảng B (2 vòng for)
  - Nếu ô đó không chứa mìn (có giá trị khác -1) thì
    - Đếm số mìn trong 8 ô xung quanh nó (chú ý các ô nằm ở biên có thể không có đủ 8 ô xung quanh)



Đếm số mìn

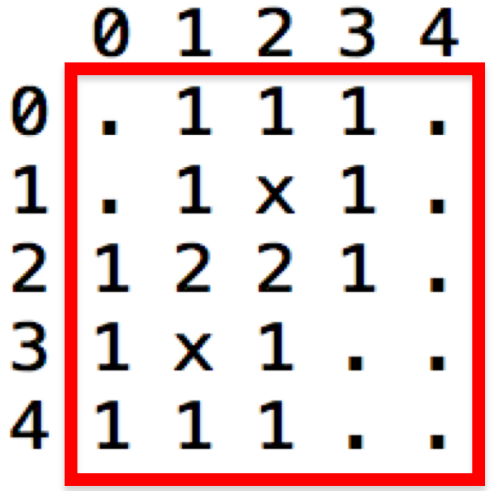
- Một phần đoạn code tính số mìn xung quanh 1 ô được minh hoạ bên dưới




```
void count_mines() {
    for (int i = 0; i < M; i++)
        for (int j = 0; j < N; j++)
            if (B[i][j] != MINE) {
                int cnt = 0;
                if (i-1 >= 0 && j-1 >= 0 && B[i-1][j-1] == MINE)
                    cnt = cnt + 1;
                if (i-1 >= 0 && B[i-1][j] == MINE)
                    cnt = cnt + 1;
                if (i-1 >= 0 && j+1 < N && B[i-1][j+1] == MINE)
                    cnt = cnt + 1;
```

In bản đồ mìn

- Viết hàm printMap1() để in bản đồ mìn theo quy tắc:
  - Nếu ô B[i][j] == -1 (mìn), in ký tự 'x'
  - Nếu ô B[i][j] == 0, in ký tự '.'
  - Ngược lại in giá trị của B[i][j]



- Chú ý:
  - Cần phải in cả chỉ số hàng và chỉ số cột
  - Mỗi ô cách nhau 1 khoảng trắng

In bản đồ mìn

- Hàng đầu tiên (chỉ số cột)
  - In 2 khoảng trắng
  - In chỉ số cột từ 0 đến N-1
  - Xuống dòng
- Các hàng còn lại (từ 0 đến M-1)
  - In chỉ số cột
  - In các giá trị trong cột theo quy tắc
  - Xuống dòng

```
  0 1 2 3 4
0 . 1 1 1 .
1 . 1 x 1 .
2 1 2 2 1 .
3 1 x 1 . .
4 1 1 1 . .
```

In bản đồ mìn

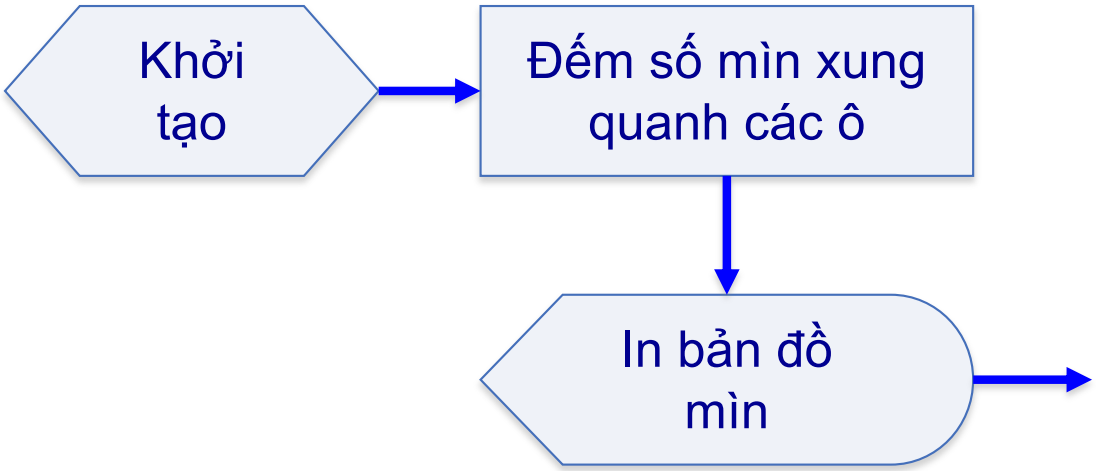
```
void printMap1() {
    printf("  "); /*2 khoảng trắng*/
    for (int j = 0; j < N; j++)
        printf("%d ", j%10); /*chỉ số cột*/
    printf("\n"); /*xuống dòng*/

    for (int i = 0; i < M; i++) {
        printf("%d ", i%10); /*chỉ số hàng*/
        for (int j = 0; j < N; j++)
            if (B[i][j] == MINE)
                printf("x ");
            else if (B[i][j] == 0)
                    printf(". ");
            else printf("%d ", B[i][j]);
        printf("\n"); /*xuống dòng*/
    }
}
```

```
  0 1 2 3 4
0 . 1 1 1 .
1 . 1 x 1 .
2 1 2 2 1 .
3 1 x 1 . .
4 1 1 1 . .
```

Bài tập 1

- Đến lúc này ta đã có các thành phần
  - Khai báo
  - Khởi tạo
  - Đếm số mìn
  - In bản đồ mìn
- Lắp ráp các phần trên để có được một chương trình thực thi công việc sau:
  - Khởi tạo bản đồ mìn với 2 quả mìn ở vị trí (1, 2) và (3, 1)
  - Đếm số mìn xung quanh các ô
  - In bản đồ mìn

Lưu đồ chương trình & hàm main()




```
int main() {
        init();
        count_mines();
        printMap1();
        return 0;
}
```

Thực hành

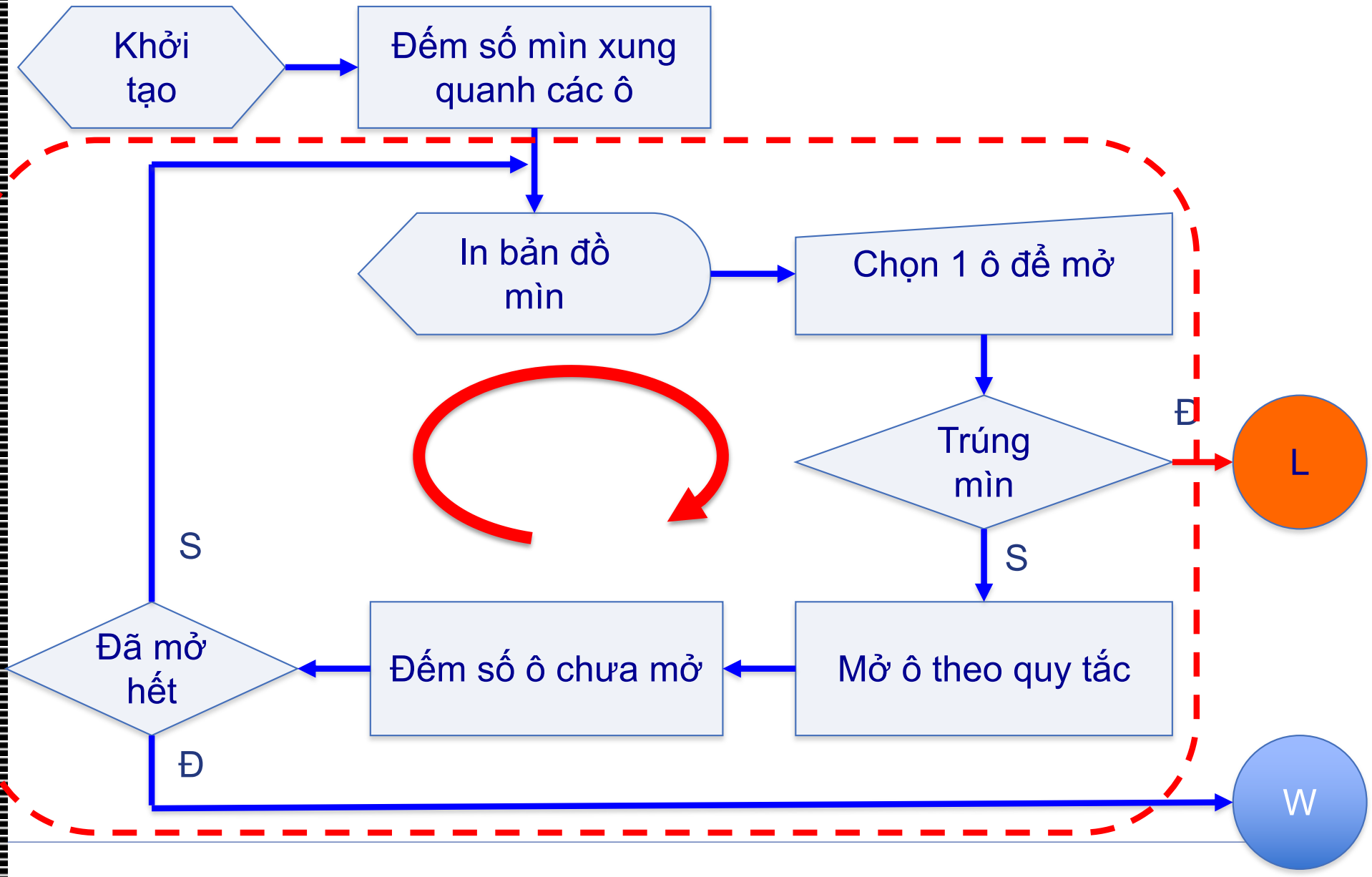
```
  0 1 2 3 4
0 . 1 1 1 .
1 . 1 x 1 .
2 1 2 2 1 .
3 1 x 1 . .
4 1 1 1 . .
```

- Yêu cầu:
  - Lắp ráp các thành phần trên (kể cả hàm main) thành chương trình hoàn chỉnh
  - Biên dịch và chạy thử
  - Nếu làm đúng kết quả sẽ như hình trên

Bài tập 2

- Hoàn chỉnh trò chơi SimpleMinesweeper

Lưu đồ tổng quát



Khung chương trình

```
int main() {
        int r, c;
        int k = 2;
        init();
        count_mines();
        while (1) {
                /*Chọn ô cần mở*/
                /*Kiểm tra trúng mìn*/
                /*Mở ô (r,c)*/
                /*Đếm ô chưa mở*/
                /*Kiểm tra kết thúc*/
        }
        return 0;
}
```

Chọn ô để mở

- Người dùng chọn ô cần mở
  - Yêu cầu người dùng nhập vào hàng và cột ô cần mở, lưu hàng vào biến **r** và cột vào biến **c**. Để đơn giản có thể giả sử người dùng luôn nhập đúng

```
printf("Nhập ô cần mở: ");
scanf("%d%d", &r, &c);
```

Kiểm tra "trúng mìn"

- Nếu ô được chọn có mìn B[r][c] == MINE
  - Mở tất cả các ô
  - In bản đồ mìn
  - In ra câu "Bạn đã thua"
  - Thoát

```
if (B[r][c] == MINE) {
        /*Mở tất cả các ô*/
        /*In bàn cờ*/;
        printf("You lose.\n");
        break;
}
```

Mở ô theo quy tắc

- Để đơn giản hoá trò chơi, ta lần lược xét 3 “quy tắc mở ô” từ dễ đến khó như sau:
  - Quy tắc 1: chỉ mở ô (r, c) không xét các ô lân cận của nó
  - Quy tắc 2: mở ô (r, c), nếu xung quanh ô (r, c) không có mìn (B[r][c] == 0) xét để mở 8 ô xung quanh nó
  - Quy tắc 3: mở ô (r, c), nếu xung quanh ô (r, c) không có mìn (B[r][c] == 0), áp dụng quy tắc 3 lên 8 ô xung quanh => đệ quy

Quy tắc 1

- Viết hàm open_cell_1(int r, int c) để mở ô (r, c) theo quy tắc 1:
  - Chuyển trạng thái của ô (r, c) từ chưa mở (0) sang đã mở (1)

```
void open_cell_1(int r, int c) {
        if (T[r][c] == 0)
                T[r][c] = 1;
}
```

Đếm số ô chưa mở

- Viết hàm int count_remain() để đếm số ô chưa mở (số ô còn lại):
  - Đặt cnt = 0;
  - Duyệt qua từng ô, nếu T[i][j] == 0 tăng cnt lên 1
  - Trả về cnt

```
int count_remain() {
        int cnt = 0;
        for (int i = 0; i < M; i++)
                for (int j = 0; j < N; j++)
                        if (T[i][j] == 0)
                                cnt++;
        return cnt;
}
```

Kiểm tra kết thúc

- Trò chơi kết thúc khi sô ô chưa mở còn lại đúng bằng k

```
if (count_remain() == k) {
        printf("Bạn thắng rồi !\n");
        /*kết thúc*/
        break;
}
```

In bản đồ mìn theo trạng thái

- Viết lại hàm void printMap2() in bản đồ mìn theo quy tắc sau (có quan tâm đến trạng thái của các ô):
  - Nếu ô chưa mở => in ký tự '#'
  - Ngược lại
    - Nếu ô đó chứa mìn => in 'x'
    - Nếu xung quanh không có mìn => in '.'
    - Ngược lại => in giá trị B[i][j] của ô đó

Thực hành

- Lắp ráp các thành phần trên thành chương trình hoàn chỉnh

```
int main() {
    int r, c;
    k = 2;
    init();
    count_mines();
    while (1) {
        printMap2();
        printf("Nhập ô cần mở: "); scanf("%d%d", &r, &c);

        if (B[r][c] == MINE) {
            openAll();
            printMap2();
            printf("You lose.\n");
            break;
        }
        open_cell_1(r, c);
        if (count_remain() == k) {
            printMap2();
            printf("You win.\n");
            break;
        }
    }
    return 0;
}
```

Bài tập nâng cao

- Cài đặt hàm open_cell_2(r, c) để mở ô (r, c) theo quy tắc 2
- Cài đặt hàm open_cell_3(r, c) để mở ô (r, c) theo quy tắc 3
  - Sử dụng đệ quy (xem hướng dẫn)
- Cài đặt hàm init_mines(int k) để khởi tạo bản đồ và đặt k quả mìn ở các vị trí ngẫu nhiên
  - Sử dụng hàm rand() để sinh ra số ngẫu nhiên
- Cải tiến hàm đếm số mìn xung quanh và gộp chung nó với hàm init_mines(int k)

Mở ô theo quy tắc 3

- Gọi ô cần mở là (r, c)
  - Nếu (r < 0 || r >= M || c < 0 || c >= N) => nằm ngoài bản đồ, kết thúc
  - Nếu ô này đã mở => không cần mở nữa, kết thúc
  - Nếu ô này có B[r][c] > 0 => xung quanh nó có mìn, không mở các ô xung quanh
  - Ngược lại (B[r][c] == 0), gọi đệ quy mở 8 ô xung quanh nó
- Chú ý:
  - Ta luôn có thể giả sử B[r][c] != MINE (vì đã kiểm tra rồi) nên không cần kiểm tra trường hợp này

Hàm init_mines(int k)

- Sử dụng hàm rand() để sinh ra một số ngẫu nhiên
- Phương pháp khởi tạo ngẫu nhiên k quả mìn
  - Bước 0:
    - khởi tạo tất cả các ô B[i][j] = 0
    - gán số mìn đã đặt cnt = 0
  - Bước 1: Sinh 1 số ngẫu nhiên từ 0 đến M – 1 gán vào r
    - r = rand()%M;
  - Bước 2: Sinh 1 số ngẫu nhiện từ 0 đên N – 1 gán vào c
    - c = rand()%N;
  - Bước 3: Nếu ô (r, c) chưa có mìn
    - Đặt quả mìn vào ô này: B[r][c] = MINE;
    - Tăng số mìn đã đặt lên 1: cnt++;
  - Lặp lại Bước 1 cho đến khi số mìn đã đặt = k

Đếm số mìn xung quanh các ô

- Quan sát:
  - Nếu bao quanh các ô chứa mìn bằng một hình vuông kích thước 3 x 3 thì:
    - Các ô không có hình vuông nào phủ lên có giá trị 0
    - Ô có 1 hình vuông phủ lên có giá trị 1
    - Ô có 2 hình vuông phủ lên có giá trị 2
- Nhận xét:
  - Giá trị của mỗi ô = số hình vuông phủ lên nó.

	0	1	2	3	4
0	0	1	1	1	0
1	0	1	-1	1	0
2	1	2	2	1	0
3	1	-1	1	0	0
4	1	1	1	0	0

Đếm số mìn xung quanh các ô

- Kết hợp việc đếm số mìn với việc khởi tạo vị trí các quả mìn
- Mỗi lần đặt 1 quả mìn (bước 3):
  - Tăng giá trị B[i][j] của các ô xung nó lên 1.

	0	1	2	3	4
0	0	1	1	1	0
1	0	1	-1	1	0
2	1	2	2	1	0
3	1	-1	1	0	0
4	1	1	1	0	0

CT101 – Lập trình căn bản

Khoa CNTT&TT – ĐHCT