```
/*
 * advanced programming
 * Alireza Akhavan Pour
 * akhavan@alirezaWeb.com
 * date: 1395/02/04
 */
int main()
{
      cout<< "In the name of God";
      Lecture_19();
      return 0;
}

string Lecture_19()
{
      cout << “1.Robofight!" <<endl;
      return ":)"
}
```

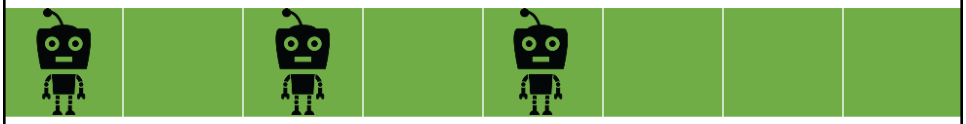
Shahid Rajaee Teacher Training University

1

2

The Robofight game

✓ a number of dumb fighter robots on a linear playground (called Roboline)



3

قوانین بازی

❑در هر نوبت (turn):

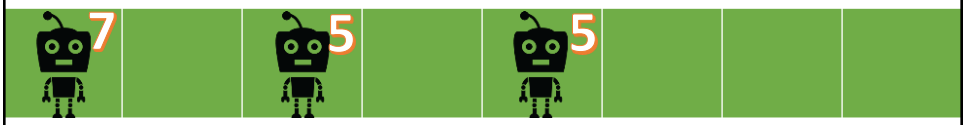
✓هر ربات به صورت رندم یک قدم به چپ یا راست می رود.

✓اگر با ربات دیگری برخورد کرد، تا سرحد مرگ با هم میجنگند (یکی باید بیمرد!)

❑هر ربات (staminaطاقت یا به اصطلاح جان) متفاوت دارد.

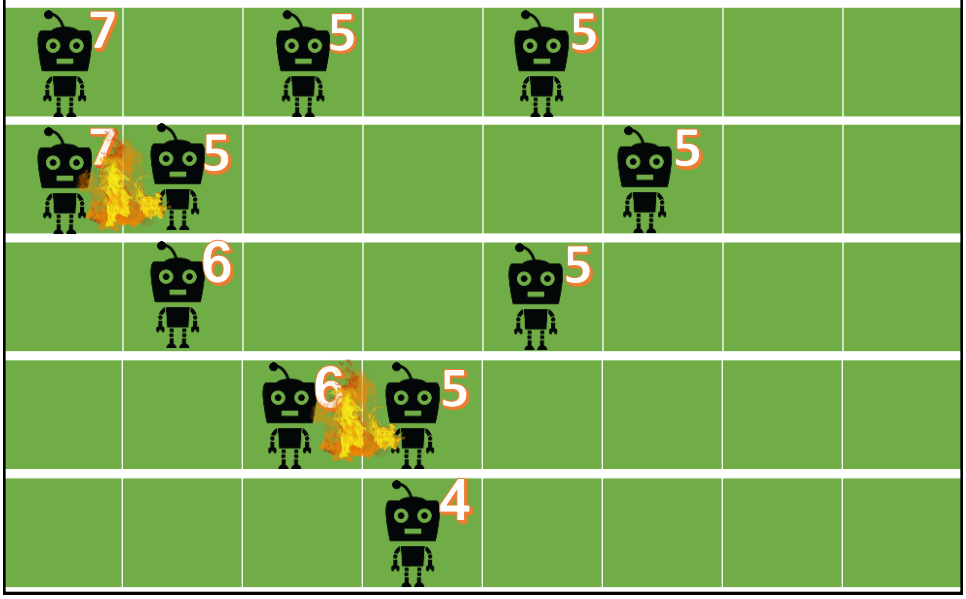
✓هر جنگ یک واحد از stamina کم میکند.

✓برنده از staminaی ربات مشخص میشود.



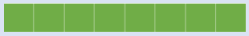
4

روبوفایت در عمل:



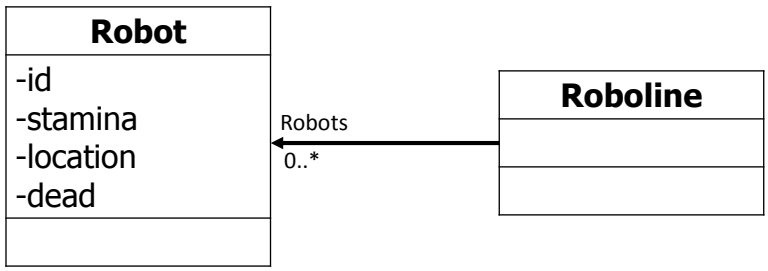
کلاس‌های برنامه

Robot	knows	Does
	It's stamina It's location	fighting moving

Roboline	knows	Does
	It's length The robots in the game	creating robots giving turn destroying robots

6

Program structure - 1



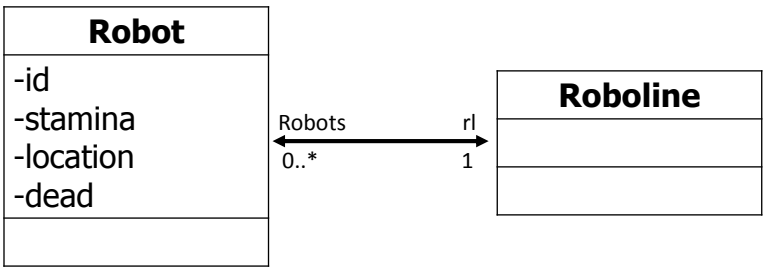
7

A turn in robots life

- determine your destination
- if the destination is outside Roboline
  - stay where you are
- if the destination is onoccupied
  - go to the destination
- else
  - fight with the robot at destination
  - if you are not dead
    - go to the destination

8

Program structure - 2



9

Roboline::turn()

```
void Roboline::turn() {
    for (int i = 0; i < robots.size(); i++)
        robots[i]->play_turn();
}
```

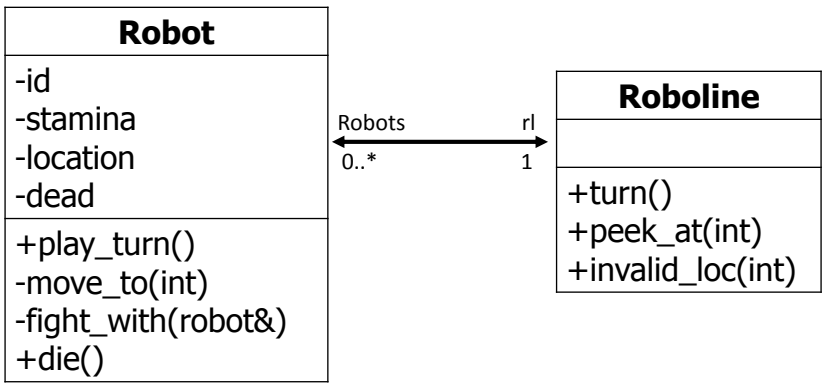
10

Robot::play_turn()

```
destination = ...

if (rl->invalid_loc(destination))
    return;

Robot* other = rl->peek_at(destination);

if (other == NULL)
    move_to(destination);
else {
    fight_with(*other);
    if (!dead)
        move_to(destination);
}
```

11

UML Class Diagram

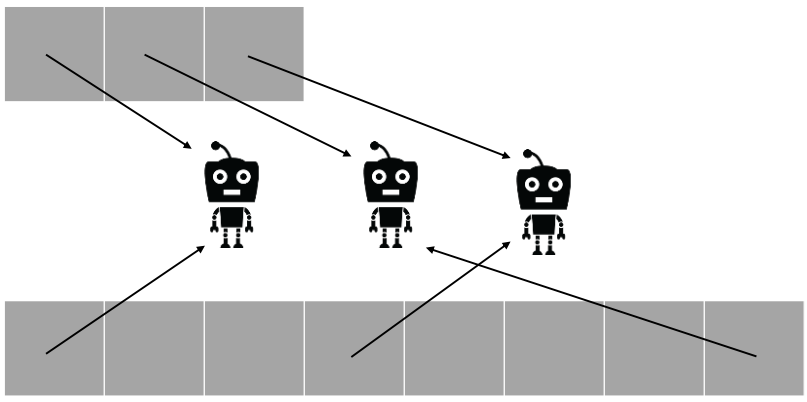


12

سوالات

- چگونه Roboline::peek_at را پیاده سازی کنیم؟
- اگر ربات بمیرد چه میشود؟
- کی برنامه را تمام کنیم؟

13



14

کد:

- در این بخش یک رقابت خیالی بین روبات‌ها شبیه‌سازی می‌شود. صورت مسئله به این شکل است که تعدادی روبات در یک محوطه‌ی خطی قرار گرفته‌اند و به نوبت بازی می‌کنند. در هر نوبت، یک روبات به طور تصادفی تصمیم می‌گیرد یک خانه به سمت راست یا چپ برود. در صورتی که خانه‌ی مقصد از ابعاد محوطه بیرون باشد روبات حرکت نمی‌کند. در غیر این صورت، اگر خانه‌ی مقصد خالی بود روبات به آن خانه نقل مکان می‌کند، اما اگر روبات دیگری در آن خانه باشد بین آن دو جنگ در می‌گیرد. هر روبات انرژی مشخصی دارد که تا حدی تعیین کننده‌ی نتیجه‌ی جنگ است. برنده‌ی جنگ، به طور تصادفی اما با وزن متناسب با انرژی روبات‌ها تعیین می‌شود. پس از جنگیدن، روبات بازنده می‌میرد و از محوطه حذف می‌شود. در صورتی که روباتی که قصد حرکت داشته برنده‌ی جنگ باشد، به خانه‌ی مورد نظر خود منتقل می‌شود. بر اثر خستگی، روبات برنده یک واحد انرژی خود را از دست می‌دهد و اگر انرژی‌اش صفر شد او نیز از بین می‌رود. روبات‌ها به ترتیب نوبت بازی می‌گیرند تا زمانی که کم‌تر از دو روبات باقی بماند.

1_robofight.cpp

15

```
#include <iostream>
#include <string>
#include <vector>
#include <cstdlib>
#include <ctime>
using namespace std;

#define DIR_LEFT 0
#define DIR_RIGHT 1
#define MAX_POWER 9

class Roboline;

class Robot {
public:
    Robot(char i, int stm, int loc, Roboline* line)
        : stamina(stm), location(loc), dead(false), rl(line), id(i) {}

    virtual void play_turn();
    void die();

protected:
    void move_to(int dest);
    void get_tired();
    void fight_with(Robot&);

    char id;
    int stamina;
    int location;
    bool dead;
    Roboline *rl;

    friend ostream& operator<<(ostream&, const Robot&);
};
```

ثابت تعریف شده برای جهت حرکت راست

بیشترین انرژی ممکن برای روبات‌ها

چون دو کلاس Robot و Roboline به یکدیگر ارجاع متقابل دارند لازم است وجود کلاس Roboline را به این شکل اعلان کنیم.

کلاسی که روبات‌ها را توصیف می‌کند

متدی که رفتار یک روبات را در یک دور بازی آن پیاده‌سازی می‌کند. برای این که زیرکلاس‌ها بتوانند این رفتار را بازنویسی کنند، این متد مجازی تعریف شده است.

متد کمکی برای مردن روبات

متد کمکی برای نقل مکان روبات به یک خانه‌ی جدید

متد کمکی برای کم کردن انرژی روبات پس از جنگ

متد کمکی برای جنگ روبات با یک روبات دیگر

یک کاراکتر که شناسه‌ی روبات است

انرژی روبات

مکان روبات در محوطه

آیا روبات مرده؟

اشاره‌گر به محوطه

16

```
class Roboline {
public:
    Roboline(int size, int num_of_robots);

    bool more_than_one_robot();
    Robot* peek_at(int loc);
    bool invalid_loc(int loc);
    void robot_moved(int from, int to);
    void robot_died_at(int loc);
    void remove_all();
    void turn();

private:
    vector<Robot*> robots;
    vector<Robot*> line;

    friend ostream& operator<<(ostream&, const Roboline&);
};
```

کلاسی که محوطه‌ی رقابت را مدل می‌کند

آیا بیش از یک روبات در محوطه وجود دارد؟

روباتی که در خانه‌ی loc است را برگردان

آیا loc یک بیرون محوطه قرار دارد

روبات خانه‌ی from به to نقل مکان کرد

روبات خانه‌ی loc مرد

تمام روبات‌ها را از بین ببر

یک دور به همه نوبت بازی بده

مجموعه‌ی تمام روبات‌ها

محوطه: line[i] خانه‌ی i از محوطه است:
در صورتی که روباتی در آن باشد اشاره‌گر به روبات وگرنه NULL خواهد بود.


```
void Robot::play_turn() {
    if (dead)
        return;
    int dir = rand() % 2;
    int destination = location;
    if (dir == DIR_LEFT)
        destination = location - 1;
    else if (dir == DIR_RIGHT)
        destination = location + 1;

    if (rl->invalid_loc(destination))
        return;

    Robot* other = rl->peek_at(destination);

    if (other == NULL)
        move_to(destination);
    else {
        fight_with(*other);
        if (!dead)
            move_to(destination);
    }
}
```

```
void Robot::die() {
    dead = true;
    rl->robot_died_at(location);
}

void Robot::move_to(int dest) {
    rl->robot_moved(location, dest);
    location = dest;
}

void Robot::get_tired() {
    stamina--;
    if (stamina == 0)
        die();
}

void Robot::fight_with(Robot& j) {
    int r = rand() % (stamina + j.stamina);
    if (r < stamina) {
        j.die();
        get_tired();
    } else {
        die();
        j.get_tired();
    }
}
```

19

```
ostream& operator<<(ostream& out, const Robot& j) {
    out << j.id << j.stamina;
    return out;
}

Roboline::Roboline(int size, int num_of_robots)
        : line(size)
{
    for (int i = 0; i < num_of_robots; i++) {
        int loc = rand() % size;
        while (line[loc] != NULL)
            loc = rand() % size;

        Robot* new_robot = new Robot('a'+i, rand() % MAX_POWER + 1, loc, this);

        robots.push_back(new_robot);
        line[loc] = new_robot;
    }
}
```

line را در ابتدا به برداری به اندازه‌ی محوطه مقداردهی می‌کنیم.

ایجاد یک روبات در heap

20

```
bool Roboline::invalid_loc(int loc) {
    return (loc < 0) || (loc >= line.size());
}

Robot* Roboline::peek_at(int loc) {
    if ((loc >= 0) && (loc < line.size()))
        return line[loc];
    else return NULL;
}

void Roboline::robot_moved(int from, int to) {
    line[to] = line[from];
    line[from] = NULL;
}

void Roboline::robot_died_at(int loc) {
    line[loc] = NULL;
}

void Roboline::remove_all() {
    for (int i = 0; i < robots.size(); i++)
        delete robots[i];
}
```

حذف روبات‌ها از heap

21

```
bool Roboline::more_than_one_robot() {
    bool seen_any = false;
    for (int i = 0; i < line.size(); i++)
        if (line[i] != NULL)
            if (seen_any)
                return true;
            else
                seen_any = true;
    return false;
}

void Roboline::turn() {
    for (int i = 0; i < robots.size(); i++)
        robots[i]->play_turn();
}

ostream& operator<<(ostream& out, const Roboline& rl) {
    for (int i = 0; i < rl.line.size(); i++)
        if (rl.line[i] == NULL)
            out << "__ ";
        else
            out << *(rl.line[i]) << ' ';
    return out;
}
```

22

```
int main() {
    srand(time(0));
    Roboline rl(15, 5);

    while (rl.more_than_one_robot()) {
        cout << rl << endl;
        rl.turn();
        cin.get();
    }
    cout << rl << endl;
    rl.remove_all();
}
```

مقداردهی seed تولید اعداد تصادفی با زمان فعلی سیستم

23

منابع:

- http://ramtung.ir/apnotes/html/12_Robofight.html

24