Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Кафедра ПМ иК

**Курсовая работа**

по дисциплине «Объектно-ориентированное программирование»

Тема: «Игра Жизнь (4 вариант)»

Выполнил:студент группы ИВ-222

Рудцких Владислав Евгеньевич

Проверил: доцент кафедры ПМиК

Ситняковская Е.И.

Новосибирск 2023

**ОГЛАВЛЕНИЕ**

[ЗАДАНИЕ 1](#_Toc793021083)

[ТЕКСТ ПРОГРАММЫ 2](#_Toc1979809535)

[РЕЗУЛЬТАТ РАБОТЫ 3](#_Toc312166890)

# **Постановка задачи**

*Игроку предоставляется игровое поле размером 10 клеток на 10 клеток*

*(реализованное в консоли). Каждый определенный период времени*

*(например, 2 секунды) или по нажатию клавиши сменяется жизненный цикл.*

*На клетках находятся живые организмы, определенных типов:*

* *Тип Растение: является пищей для травоядного животного; не*

*двигается, появляется в незанятой клетке в случайном порядке раз в*

*несколько жизненных циклов.*

* *Тип Травоядное животное: поглощает растения и является пищей для*

*хищника; каждое травоядное двигается на случайную клетку*

*поблизости раз в несколько жизненных циклов (период должен*

*выбираться случайным образом), с некоторой вероятностью оставляя*

*после себя травоядное животное. Наступая на клетку с растением*

*поглощает его.*

* *Тип Хищник: поглощает травоядных, но не взаимодействует с*

*растениями; каждый хищник двигается на случайную клетку*

*поблизости раз в несколько жизненных циклов, поглощая всех*

*травоядных, которых встретит; с растениями не взаимодействует (или*

*взаимодействует, если это прописано в варианте задания, которые*

*указаны ниже). Способ размножения указывается в варианте.*

*Травоядные и хищники обладают системой голода. Изначально у*

*каждого из травоядных и хищников показатель голода равен определенному*

*значению, указанному в вариантах ниже. Каждое перемещение живого*

*организма отнимает 0,2 от показателя голода. Рождение нового живого*

*организма отнимает 0,4. Поглощение другого вида прибавляет к голоду 0,2.*

*В случае, когда состояние голода станет равно нулю – живое существо*

*погибает.*

*Растения живут ограниченное количество циклов, а затем погибают.*

*Растение – брусника, травоядное животное – хомяк,*

*хищник – сокол. Хомяк размножается с вероятностью в 25%. Хомяки имеют*

*показатель голода 1. Соколы имеют показатель голода 1,5. Брусника*

*появляется очень на поле очень часто и живет 10 жизненных циклов.*

*Соколы перемещаются каждый жизненный цикл вправо и вверх по*

*диагонали. Если встречается клетка с брусникой, то они перелетают ее и*

*встают на следующую клетку. На поле всегда находится минимум 3 сокола.*

*Достигая границы поля, сокол появляется на противоположной его границе с*

*тем же показателем голода. При размножении хомяков их появляется по 2, а*

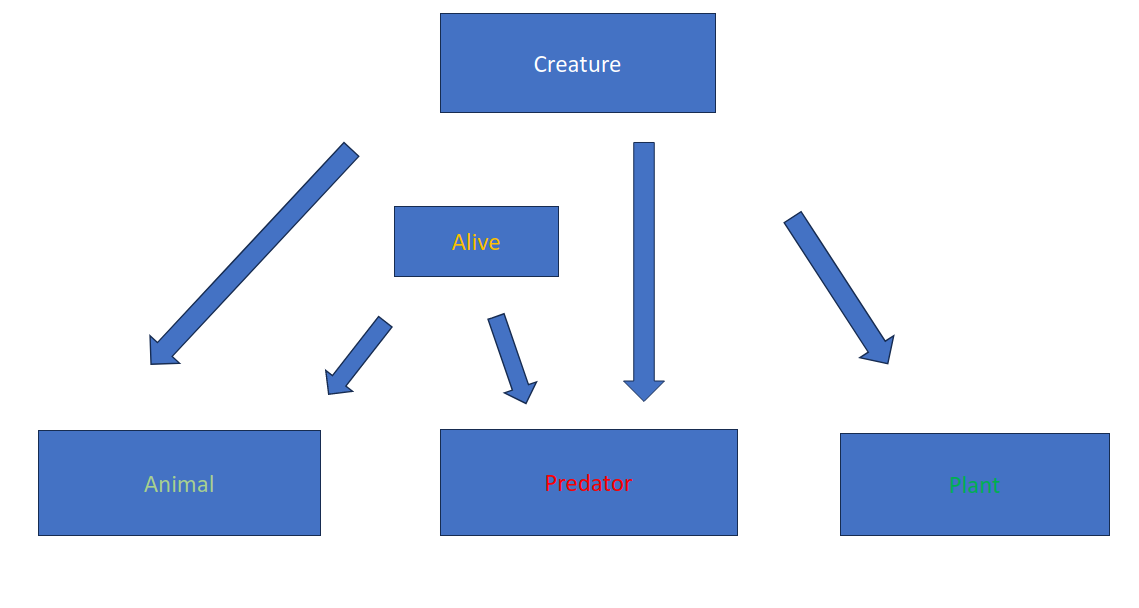
*не по 1. Сокол не может появиться в углу поля.*

Технологии ООП

В данной курсовой работе применялось множество технологий ООП, например такие как:

* Наследование: В программе существует своя иерархия классов, где Creature является классом родителем для трех других
* Полиморфизм: Классы потомки перегружают конструктор класса родителя, а также несколько методов
* Инкапсуляция: Поля классов ограничены классом и в некоторых случаях наследуются от класса Creature
* Абстракция: Реализовано два абстрактных класса Creature и Alive которые позволяют назначить определенные методы присущие этим классам для перегрузки

Структура классов



Программная реализация

#include <iostream>

#include <vector>

#define stay 1

#define flying 2

#define moving 3

#define TABLESIZE 10

#define ran (rand() % TABLESIZE)

#ifdef \_WIN32

#include <Windows.h>

#include <stdlib.h>

#endif

#ifdef \_\_linux\_\_

#include <stdlib.h>

#include <unistd.h>

#define Sleep(x) sleep((x/1000))

#define system(cls) system("clear")

#endif

using std::vector;

using namespace std;

class Alive {

protected:

double hunger = 0;

public:

virtual void changeHunger(double amount) = 0;

};

class Creature {

protected:

int x = 0, y = 0;

char state = 0;

char symbol = 0;

public:

int getX() {

return x;

}

int getY() {

return y;

}

void setX(int a){

x = a;

}

void setY(int b) {

y = b;

}

virtual void spawn(char\*\* table, int a, int b) = 0;

virtual void action(char\*\* table) = 0;

virtual void printinfo(const char\* name) {

cout << "obj: " << name << " | X:" << this->x << " Y:" << this->y << endl;

}

};

class Plant : public Creature {

char symbol = '\*';

char state = stay;

int timeleft = 10;

int living = 1;

public:

int num = 0;

void timepass() {

this->timeleft--;

}

Plant() {

x = rand() % (TABLESIZE);

y = rand() % (TABLESIZE);

}

Plant(int a, int b) {

x = a;

y = b;

}

int isAlive() {

return living;

}

void spawn(char\*\* table, int a, int b) {

x = a;

y = b;

table[x][y] = symbol;

}

void spawn(char\*\* table) {

while (table[x][y] != '#') {

this->x = rand() % (TABLESIZE);

this->y = rand() % (TABLESIZE);

}

table[x][y] = symbol;

}

int getNum() {

return num;

}

void setNum(int n) {

num = n;

}

void action(char\*\* table) {

timeleft--;

if (timeleft == 0) {

table[this->x][this->y] = '#';

living = 0;

}

if (table[x][y] != '\*'){

living = 0;

table[x][y] = '#';

}

}

};

class Predator : public Creature, Alive {

char symbol = '@';

char state = flying;

int living = 1;

public:

Predator() {

hunger = 1.5;

x = rand() % (TABLESIZE);

y = rand() % (TABLESIZE);

}

Predator(int a, int b) {

x = a;

y = b;

}

void changeHunger(double amount) {

this->hunger += amount;

}

void spawn(char\*\* table,int a,int b) {

x = a;

y = b;

table[x][y] = symbol;

}

void action(char\*\* table) {

int corner = 0;

table[x][y] = '#';

x++; y++;

if ((x >= TABLESIZE) || (y >= TABLESIZE)) {

x = 0; y = 0;

table[x][y] = symbol;

}

if (table[x][y] == '\*'){

x++; y++;

}

if ((x > TABLESIZE) && (y > TABLESIZE))

corner = 1;

if (x > TABLESIZE) {

x = 0;

y = y - 1;

} else if (y > TABLESIZE) {

x = x - 1;

y = 0;

}

if (corner == 1) {

x = 0;

y = 0;

}

if (table[x][y] == '%')

changeHunger(0.2);

table[x][y] = symbol;

}

};

class Animal : public Creature, Alive {

char symbol = '%';

char state = moving;

int living = 1;

public:

Animal() {

hunger = 1.0;

x = rand() % (TABLESIZE + 1);

y = rand() % (TABLESIZE + 1);

}

Animal(int a, int b) {

x = a;

y = b;

}

int isAlive() {

return this->living;

}

double getHunger() {

return this->hunger;

}

void changeHunger(double amount) {

this->hunger += amount;

}

void spawn(char\*\* table,int a,int b) {

x = a; y = b;

if(table[x][y]=='#')

table[x][y] = symbol;

//changeHunger(-0.4);

}

void action(char\*\* table) {

int moveX, moveY;

int count = 10;

if (table[x][y] != '%')

living = 0;

//while ((table[x][y] != '#') || (table[x][y] != '\*') || (table[x][y]==NULL)) {

do{

if (count == 0)

break;

moveX = x;

moveY = y;

moveX += ((rand() % 3) - 1);

moveY += ((rand() % 3) - 1);

count--;

}while (table[x][y]!='#');

if (count != 0) {

table[x][y] = '#';

x = moveX;

y = moveY;

if (table[x][y] == '\*')

changeHunger(0.2);

table[x][y] = '%';

}

}

};

char\*\* createTable(int x, int y) {

char\*\* table = new char\*[x];

for (int i = 0; i < x; i++)

table[i] = new char[y];

for (int i = 0; i < x; i++) {

for (int j = 0; j < y; j++)

table[i][j] = '#';

}

return table;

}

void printTable(char\*\* table) {

int x, y;

x = TABLESIZE;

y = TABLESIZE;

for (int i = 0; i < x; i++) {

cout << endl;

for (int j = 0; j < y; j++)

cout << table[i][j] << ' ';

}

}

int main() {

srand(time(0));

char\*\* Table = createTable(10,10);

int time = 1000;

std::vector<Plant\*> plants;

std::vector<Animal\*> animals;

std::vector<Predator\*> predators;

Plant\* newPlant = new Plant();

newPlant->spawn(Table);

plants.push\_back(newPlant);

for (int i = 0; i < 3; i++) {

Predator\* newPredator = new Predator();

newPredator->spawn(Table,ran,ran);

predators.push\_back(newPredator);

}

Animal\* newAnimal = new Animal();

newAnimal->spawn(Table,ran,ran);

animals.push\_back(newAnimal);

int chance = 0;

while (time != 0) {

time--;

printTable(Table);

for (long unsigned int i = 0; i < plants.size();i++)

plants[i]->action(Table);

chance = (rand() % 4);

if (chance == 0) {

newAnimal->changeHunger(-0.4);

for (int i = 0; i < 2; i++) {

Animal\* newAnimal = new Animal();

newAnimal->spawn(Table, ran, ran);

animals.push\_back(newAnimal);

}

}

chance = rand() % 2;

if (chance == 0){

Plant\* newPlant = new Plant();

newPlant->spawn(Table);

plants.push\_back(newPlant);

}

for (long unsigned int i = 0; i < predators.size(); i++)

predators[i]->action(Table);

for (long unsigned int i = 0; i < animals.size(); i++)

animals[i]->action(Table);

Sleep(1\*1000);

system("cls");

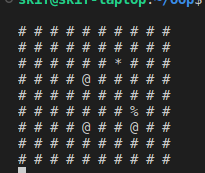
}

return 0;

}

**Результаты работ**

О



# **РЕЗУЛЬТАТ РАБОТЫ**

Ис