Отчёт

Задача 100

Подготовил Марченко С.И. КВБО-01-21

**Программа «Жизнь», (описывающая жизнь микроорганизмов)**

**Введение:**

Жили-были микробы. Жили они долго и счастливо, но вот одна беда: после жизни всегда приходит смерть. А микробы были разные: маленькие и не очень, старички и малютки – и у каждого из них был свой жизненный уровень. Так, например, у только что родившегося микробчика он был равен 1, а по мере его взросления жизненный уровень тоже рос (от 1 до 12). Когда же микроб достигал последней ступени (т.е. 12), он, увы, погибал (теперь его жизненный уровень равен 0). Если же у микроба уровень был равен 0, то он рождался заново и проходил опять жизнь бодрым шагом от 1 до 12. Всего же микробов в мире, как вы знаете, очень много, а жизнь их интересна. Поэтому появилась идея написать программу, описывающую их жизнь. Эта программа должна сообщать нам количество микробов в каждом поколении и «нарисовать» их на поле (экране).

**Программа должна работать следующим образом:**

1. Сначала создаётся файл work.dat (жилище микробов) и файл work.out (описывающий текущее поколение микробов, их развитие). Файл work.dat состоит из различных символов, среди которых «обитают» микробы.

2. Необходимо создать массивы «настоящее» и «будущее». В массиве «настоящее» записывается текущее поколение микробов, а в файл «будущее»- следующее. Массивы создаются размером 21х21.

3. Программа выдаёт пользователю на экран запрос: «Введите количество поколений». Именно столько поколений программа будет описывать.

4. Теперь программа создаёт в файле work.out поколение под номером 1. Для этого программа открывает и проверяет на наличие микробов файл (каждый символ, среди которых может затеряться микроб). Если под символом скрывается микроб (например, символ «Х»), то в массив «настоящее» записывается единица (1, т.е. микроб только что родился), а если не микроб – то ноль (0, т. е. там никто не живёт). Теперь в массиве «настоящее» находиться поле из 1 и 0 (он состоит из новорожденных младенцев и пустых мест). Все последующие поколения тоже записываются в файл work.out. Файл work.dat закрывается, и работа теперь ведётся только с файлом work.out.

5. Теперь описывается следующее поколение. Оно создаётся после проверки массива «настоящее». Проверяются микробы и их соседи (результат записывается в массив «будущее»): - если жизненный уровень микроба от 1 до 11 то: --если соседей 2 или 3, то микроб продолжает жить и подрастает (т.е. его жизненный уровень возрастает на 1), --иначе микроб погибает (=0), т.к. он задыхается или умирает от скуки; - если жизненный уровень микроба 0, то микроб рождается заново (т.е. его жизненный уровень равен 1); - если жизненный уровень микроба равен 12, то микроб погибает (от старости).

6. После проверки подсчитывается количество жизней данного поколения (т.е. сколько единиц). Затем программа проверяет, есть ли кто «живой на поле» или все погибли (т.е. нули).

7. Теперь программа производит замену поколений: массив «будущее» становится «настоящим». 100

8. Программа продолжает работу с пункта 5 до тех пор, пока количество поколений (которые описывает программа) не станет равным введенным пользователем в пункте 3 или пока не погибнут все микробы (т.е. везде одни нули).

#include <iostream>

#include <fstream>

#include <regex>

#include <map>

#include <array>

using namespace std;

char rand\_char(int n) { return ('a' + rand() % n); }

int cell\_int(char cell) { return (int)cell - 48; }

char cell\_out(char cell) { return "0123456789AB"[cell\_int(cell)]; }

int main() {

srand((unsigned)time(NULL));

cout << "Enter number of letters [1-26]: ";

int n;

cin >> n;

if (!cin || n < 1 || n > 26) return -1;

ofstream work\_dat("work.dat");

fstream work\_out("work.out");

for (int i = 0; i < 21; i++) {

for (int j = 0; j < 21; j++) {

char ch = rand\_char(n);

work\_dat << ch << " ";

}

work\_dat << endl;

}

work\_dat.close();

array<array<int, 21>, 21> present;

array<array<int, 21>, 21> future;

cout << "Enter number of generations: ";

int steps;

cin >> steps;

if (!cin) return 1;

char symbol = rand\_char(n);

ifstream fin("work.dat");

cout << endl << "Generation 0" << endl;

int c = 0;

for (int i = 0; i < 21; i++) {

char ch;

for (int j = 0; j < 21 && fin >> ch; j++) {

char cell = (ch == symbol ? '1' : 'X');

present[i][j] = cell;

work\_out << cell << " ";

cout << cell << " ";

c += cell != 'X';

}

cout << endl;

work\_out << endl;

}

fin.close();

cout << "Living cells: " << c << endl;

for (int i = 0; i < steps && c != 0; i++) {

cout << endl << "Generation " << i + 1;

c = 0;

cout << endl;

work\_out << endl;

for (int i = 0; i < 21; i++) {

for (int j = 0; j < 21; j++) {

char cell = present[i][j];

if (cell == 'X') {

work\_out << "X ";

cout << "X ";

}

else {

if (cell\_int(cell) < 11) {

int n = 0;

for (int k = max(i - 1, 0); k <= min(i + 1, 20); k++) {

for (int l = max(j - 1, 0); l <= min(j + 1, 20); l++) {

int t = cell\_int(present[k][l]);

if (!(k == i && l == j) && present[k][l] != 'X' && t >= 1) n++;

}

}

if ((cell\_int(cell) > 0 && n == 2 || n == 3) || (cell == 0 && n == 3)) {

cell++;

c++;

}

else cell = '0';

}

else cell = '0';

work\_out << cell\_out(cell) << " ";

cout << cell\_out(cell) << " ";

}

future[i][j] = cell;

}

cout << endl;

work\_out << endl;

}

cout << "Living cells: " << c << endl;

present = future;

}

if (c == 0) cout << endl << "Oh no! Everyone died!" << endl;

work\_out.close();

}

Эта строчка нужна, чтобы функция rand() возвращала случайные значения не циклично.

15 srand((unsigned)time(NULL));

Спрашиваем у пользователя необходимое кол-во букв и проверяем ввод на правильность.

17 cout << "Enter number of letters [1-26]: ";

18 int n;

19 cin >> n;

20 if (!cin || n < 1 || n > 26) return -1;

Создаём и открываем файлы work.dat для записи и work.out для записи и чтения.

22 ofstream work\_dat("work.dat");

23 fstream work\_out("work.out");

Заполняем файл work.dat случайными символами с помощью вспомогательной функции rand\_char() и закрываем его.

24 for (int i = 0; i < 21; i++) {

25 for (int j = 0; j < 21; j++) {

26 char ch = rand\_char(n);

27 work\_dat << ch << " ";

28 }

29 work\_dat << endl;

30 }

31 work\_dat.close();

Функция rand\_char() принимает мощность алфавита n и выдаёт случайную бувку в диапазоне [‘a’; ‘a’ + n).

8 char rand\_char(int n) { return ('a' + rand() % n); }

Создаём массивы настоящее и будущее.

33 array<array<int, 21>, 21> present;

34 array<array<int, 21>, 21> future;

Спрашиваем у пользователя необходимое кол-во поколений и проверяем ввод на правильность.

36 cout << "Enter number of generations: ";

37 int steps;

38 cin >> steps;

39 if (!cin) return 1;

Выбираем случайный символ, он будет значить, что тут клетка.

41 char symbol = rand\_char(n);

Открываем файл work.dat для чтения.

42 ifstream fin("work.dat");

Выводим 0 поколение.

43 cout << endl << "Generation 0" << endl;

Создаём переменную для подсчёта живых клеток.

44 int c = 0;

Проходимся по файлу work.dat и сравниваем символ в файле с выбранным нами, если они совпадают, то записываем в переменную cell клетку с возрастом 1, если нет - X (клетки нет).

45 for (int i = 0; i < 21; i++) {

46 char ch;

47 for (int j = 0; j < 21 && fin >> ch; j++) {

48 char cell = (ch == symbol ? '1' : 'X');

Затем выводим записываем полученное значение в массив настоящее и файл work.out, а также выводим значение на экран.

49 present[i][j] = cell;

50 work\_out << cell << " ";

51 cout << cell << " ";

Если клетка жива, то прибавляем 1 к счётчику.

52 c += cell != 'X';

При выводе не забываем про перенос строки. После прочтения закрываем файл work.dat и больше его не используем.

53 }

54 cout << endl;

55 work\_out << endl;

56 }

57 fin.close();

Выводим кол-во живых клеток

58 cout << "Living cells: " << c << endl;

Теперь самое интересное, симулируем поколения. Запускаем цикл по кол-ву шагов и ставим дополнительное условие – симуляция заканчивается, если все клетки мертвы.

60 for (int i = 0; i < steps && c != 0; i++) {

Выводим номер поколения, обнуляем счётчик и добавляем пустую строку на экран и в файл.

61 cout << endl << "Generation " << i + 1;

62 c = 0;

63 cout << endl;

64 work\_out << endl;

Проходимся по массиву настоящее и записываем клетку по координатам в переменную cell для удобства.

65 for (int i = 0; i < 21; i++) {

66 for (int j = 0; j < 21; j++) {

67 char cell = present[i][j];

Если на координатах нет клетки, то просто выводим X на экран и в файл.

68 if (cell == 'X') {

69 work\_out << "X ";

70 cout << "X ";

71 }

Иначе проверяем возраст клетки.

Если клетка жива и ей меньше 11 лет, то …

73 if (cell\_int(cell) < 11) {

Создаём переменную n – кол-во соседей.

74 int n = 0;

Проходимся по вертикали от i-1 до i+1, при этом надо убедиться, что мы не зайдём за границы массива [0, 20].

75 for (int k = max(i - 1, 0); k <= min(i + 1, 20); k++) {

После проходимся по горизонтали от j-1 до j+1 и так же не выходим за границы массива.

76 for (int l = max(j - 1, 0); l <= min(j + 1, 20); l++) {

Получаем числовое значение с помощью функции cell\_int() и записываем в переменную t для удобства.

77 int t = cell\_int(present[k][l]);

Функция cell\_int() принимает символ клетки и преобразует его в число от 0 до 11, если это клетка, если нет, то получается какое-то другое число, но далее есть проверка на X .

10 int cell\_int(char cell) { return (int)cell - 48; }

Проверяем клетку на соседа. Во-первых, это должна быть не та же самая клетка, чьих соседей мы проверяем. Во-вторых, это должен быть не X. В-третьих, клетка должна быть жива, чтобы считаться соседом. Если все эти условия соблюдены, то мы увеличиваем кол-во живых соседей на 1.

79 if (!(k == i && l == j) && present[k][l] != 'X' && t >= 1) n++;

Далее мы смотрим на полученное число соседей и если эта клетка жива и вокруг 2 или более соседей, то она растёт. Если клетка мертва и вокруг ровно 3 соседа, то она оживает и мы прибавляем 1 к живым клеткам.

83 if ((cell\_int(cell) > 0 && n == 2 || n == 3) || (cell == 0 && n == 3)) {

85 cell++;

86 c++;

87 }

Если ни одно из этих условий не соблюдено, то клетка умирает или остаётся мёртвой.

88 else cell = '0';

Возвращаемся к условию, что клетка должна быть жива и младше одиннадцати лет. Если это не так, то клетка умирает или остаётся мёртвой.

73 if (cell\_int(cell) < 11) {

…

89 }

90 else cell = '0';

Выводим новое значение клетки в файл и на экран с помощью функции cell\_out().

91 work\_out << cell\_out(cell) << " ";

92 cout << cell\_out(cell);

Функция cell\_out() принимает символ и возвращает его в восьмеричной системе счисления.

12 char cell\_out(char cell) { return "0123456789AB"[cell\_int(cell)]; }

Записываем новое значение в массив будущего.

94 future[i][j] = cell;

Опять же не забываем про отступы.

96 cout << endl;

97 work\_out << endl;

Выводим кол-во живых клеток и меняем массив настоящее на будущее.

99 cout << "Living cells: " << c << endl;

100 present = future;

В конце проверяем, если цикл завершился в следствие смерти всех клеток, то выводим дополнительную надпись. После закрываем файл.

102 if (c == 0) cout << endl << "Oh no! Everyone died!" << endl;

103 work\_out.close();