БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Международный институт дистанционного образования

Кафедра «Информационные системы и технологии»

**КУРСОВОЙ ПРОЕКТ**

по учебной дисциплине

«ЯЗЫКИ ПРОГРАММИРОВАНИЯ»

Исполнитель: студент 2 курса, группы 41703120 Реут В.Л.

Руководитель: Радкевич А. С.

Минск 2022

**Оглавление**

**Введение………………………………………………………………………..2**

**Задание………………………………………………………………………....3**

**Постановка задачи………………………………………………………..3**

**Действия…………………………………………………………………....4**

**Листинг программы………………………………………………………….14**

**Заключение……………………………………………………………………23**

# ВВЕДЕНИЕ

Язык C++ был разработан Бьёрном Страуструпом в 1979 г. в качестве дополнения к языку C. Он добавил множество новых возможностей в язык С. Его популярность была вызвана объектно-ориентированностью языка. C++ стал широко использоваться для разработки программного обеспечения, став одним из самых популярных языков программирования.

Язык C++ является мощным языком, которые унаследовал от С богатые возможности по работе с памятью. Поэтому нередко этот язык находит своё применение в системном программировании, а именно, при создании операционных систем, драйверов, различных утилит, антивирусов и так далее. Однако системным программированием применение данного языка не ограничивается. Нередко он также применяется для создания графических приложений, прикладных программ, игр с богатой насыщенной визуализацией, в веб-разработке или каких-либо вспомогательных сервисов, которые обслуживают веб-приложения. Таким образом, язык С++ можно назвать языком широкого пользования.

Целью данной работы: овладеть практическими навыками проектирования и разработки законченного, отлаженного и протестированного программного продукта с использованием языка высокого уровня С++; закрепить и углубить теоретические знания. Для этого были поставлены следующие задачи:

1)Реализация консольной игры “Крестики-нолики”.

# ЗАДАНИЕ

# Постановка задачи

Нашей задачей будет научить компьютер играть с пользователем в игру крестики-нолики, и научиться делать это с использованием только терминала операционной системы. Для решения поставленной задачи нам необходимо поле для игры, символ крестика, символ нолика и пустой символ, он нужен будет для отображения поля в терминале:

**Структуры и перечисления**

// Перечисление ячеек(элементов) поля

enum Cell : char// Таким образом обЪявляем чаровский тип

{

CROSS = 'X',

ZERO = '0',

EMPTY = '\_'

};

// Структура с координатами

struct Coord

{

size\_t y{ 0 };

size\_t x{ 0 };

};

// Перечисление состояний игры

enum GameProgress

{

IN\_PROGRESS,

WON\_HUMAN,

WON\_AI,

DRAW

};

#pragma pack(push, 1)

struct Game

{

Cell\*\* ppField = nullptr;// указатель на указатель ячеек поля

const uint16\_t SIZE = 5;// Поле 5х5 будет

Cell human;// Кто чем играет

Cell ai;// --//--

size\_t turn = 0;// определение хода.Четный-человек, нечетный-ИИ

GameProgress progress = IN\_PROGRESS;

};

#pragma pack(pop)

**Блок-схема игры:**



ДЕЙСТВИЯ

Опишем действия, необходимые для игры в крестики-нолики.

**Отрисовка поля в консоли:**

void \_\_fastcall drawGame(const Game& g)

{

// верхние и верхний боковой отступы,нумерация по Х

cout << endl << " ";

for (size\_t x = 0; x < g.SIZE; x++)

{

cout << x + 1 << " ";

}

cout << endl;

// само поле

for (size\_t y = 0; y < g.SIZE; y++)

{

cout << " " << y + 1 << " | ";

for (size\_t x = 0; x < g.SIZE; x++)

{

cout << g.ppField[y][x] << " | ";

}

cout << endl;

}

cout << endl << "Человек: " << g.human << endl << "Бот: " << g.ai << endl;

}

**Начало новой игры:**

void \_\_fastcall initGame(Game& g)

{

//////////////////////////////////////////////////////////////////////////////////////////

// Сделал один вложеный цикл с объявлением и инициализацией,вместо двух отдельных

g.ppField = new Cell \* [g.SIZE];

for (size\_t y = 0; y < g.SIZE; y++)// создаем строки матрицы

{

g.ppField[y] = new Cell[g.SIZE];// создаем колонки матрицы

for (size\_t x = 0; x < g.SIZE; x++)

{

g.ppField[y][x] = EMPTY;// инициализируем строки и колонки матрицы

}

}

/////////////////////////////////////////////////////////////////////////////////////////////

if (getRundomNum(0, 1000) > 500)

{

g.human = CROSS;

g.ai = ZERO;

g.turn = 0;

}

else

{

g.human = ZERO;

g.ai = CROSS;

g.turn = 1;

}

}

Внутри себя содержит условие выполняющее жеребьевку кто будет делать ход первым – ИИ или человек. Условие ссылается на функцию содержащую генератор случайных чисел:

// C++ный сложный генератор чисел

int32\_t \_\_fastcall getRundomNum(int32\_t min, int32\_t max)

{

const static auto seed = chrono::system\_clock::now().time\_since\_epoch().count();

static mt19937\_64 generator(seed);

uniform\_int\_distribution<int32\_t> dis(min, max);

return dis(generator);

}

**Координаты пользователя:**

Coord \_\_fastcall getHumanCoord(const Game& g)

{

Coord c;

do

{

cout << "Enter X (1..5): ";

cin >> c.x;

cout << "Enter y (1..5): ";

cin >> c.y;

c.x--;

c.y--;

} while (c.x > 4 || c.y > 4 || g.ppField[c.y][c.x] != EMPTY);

return c;

}

**Определение победителя:**

Происходит путем проверки условиями и циклами столбцов, строк и диагоналей на наличие соответствующих игрокам символов. Здесь же проводим проверку на ничью.

GameProgress \_\_fastcall getWon(const Game& g)

{

// Проверяем поле по строкам(Y)

for (size\_t y = 0; y < g.SIZE; y++)

{

if (g.ppField[y][0] == g.ppField[y][1] && g.ppField[y][0] == g.ppField[y][2] && g.ppField[y][0] == g.ppField[y][3] && g.ppField[y][0] == g.ppField[y][4])

{

if (g.ppField[y][0] == g.human)

{

return WON\_HUMAN;

}

if (g.ppField[y][0] == g.ai)

{

return WON\_AI;

}

}

}

// Проверяем поле по столбцам(X)

for (size\_t x = 0; x < g.SIZE; x++)

{

if (g.ppField[0][x] == g.ppField[1][x] && g.ppField[0][x] == g.ppField[2][x] && g.ppField[0][x] == g.ppField[3][x] && g.ppField[0][x] == g.ppField[4][x])

{

if (g.ppField[0][x] == g.human)

{

return WON\_HUMAN;

}

if (g.ppField[0][x] == g.ai)

{

return WON\_AI;

}

}

}

// 1ая диагональ

if (g.ppField[0][0] == g.ppField[1][1] && g.ppField[0][0] == g.ppField[2][2] && g.ppField[0][0] == g.ppField[3][3] && g.ppField[0][0] == g.ppField[4][4])

{

if (g.ppField[2][2] == g.human)

{

return WON\_HUMAN;

}

if (g.ppField[2][2] == g.ai)

{

return WON\_AI;

}

}

// 2ая диагональ

if (g.ppField[0][4] == g.ppField[1][3] && g.ppField[0][4] == g.ppField[2][2] && g.ppField[0][4] == g.ppField[3][1] && g.ppField[0][4] == g.ppField[4][0])

{

if (g.ppField[2][2] == g.human)

{

return WON\_HUMAN;

}

if (g.ppField[2][2] == g.ai)

{

return WON\_AI;

}

}

// Проверка на ничью

bool draw = true;

for (size\_t y = 0; y < g.SIZE; y++)

{

for (size\_t x = 0; x < g.SIZE; x++)

{

if (g.ppField[y][x] == EMPTY)

{

draw = false;

break;

}

}

if (!draw)

{

break;

}

}

if (draw)

{

return DRAW;

}

return IN\_PROGRESS;

}

**Координаты и алгоритм игры ИИ:**

Coord \_\_fastcall getAICoord(const Game& g)

{

////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Предвыигрыш и предпроигрыш.

for (size\_t y = 0; y < g.SIZE; y++)

{

for (size\_t x = 0; x < g.SIZE; x++)

{

if (g.ppField[y][x] == EMPTY)

{

g.ppField[y][x] = g.ai; // Гипотетический ход: комп проверяет ход на выигрышность

if (getWon(g) == WON\_AI) // если после этого хода выиграл

{

g.ppField[y][x] = EMPTY; // возвращает как было

return { y, x }; // и ходит туда

}

g.ppField[y][x] = EMPTY; // в любом случае востанавливаем в пустое(скрываем читерство бота)

g.ppField[y][x] = g.human; // Гипотетический ход от лица человека: комп ходит за человека,проверяет ход на выигрышность

if (getWon(g) == WON\_HUMAN) // если после этого хода,человек выиграл

{

g.ppField[y][x] = EMPTY; // возвращает как было

return { y, x }; // и ходит туда,тем самым мешает человеку выиграть

}

g.ppField[y][x] = EMPTY; // в любом случае востанавливаем в пустое(скрываем читерство бота)

}

}

}

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////

if (g.ppField[2][2] == EMPTY)

{

return { 2, 2 };

}

Coord arr[20];

size\_t num = 0;

if (g.ppField[0][0] == EMPTY) // Ищем все и сохраняем свободные углы

{

arr[num++] = { 0, 0 }; // сначало NUM используется(0), затем увеличивается на 1.

}

if (g.ppField[0][4] == EMPTY)

{

arr[num++] = { 0, 4 }; // NUM = 1(после использования = 2 и тд).

}

if (g.ppField[4][4] == EMPTY)

{

arr[num++] = { 4, 4 };

}

if (g.ppField[4][0] == EMPTY)

{

arr[num++] = { 4, 0 };

}

if (num > 0) // если есть свободные углы

{

const size\_t index = getRundomNum(0, 1000) % num; // случайный индекс свободного угла

return arr[index];

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Не углы.Цикл вместо ифов

for (size\_t y = 0; y < g.SIZE; y++)

{

for (size\_t x = 0; x < g.SIZE; x++)

{

if (g.ppField[y][x] == EMPTY && g.ppField[y][x] != g.ppField[0][0] && g.ppField[y][x] != g.ppField[0][4] && g.ppField[y][x] != g.ppField[4][4] && g.ppField[y][x] != g.ppField[4][0])

{

arr[num++] = { y, x };

}

}

}

if (num > 0) // если есть свободные не углы

{

const size\_t index = getRundomNum(0, 1000) % num; // случайный индекс свободного не угла

return arr[index];

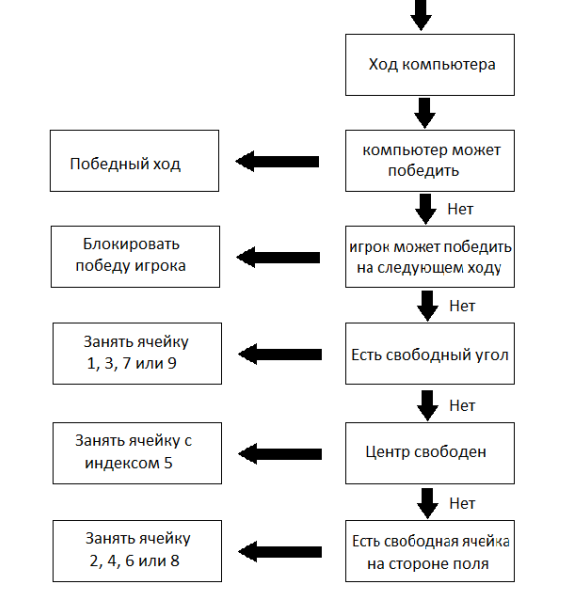
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

}

**Блок – схема ИИ:**

Схема отображает действия на поле 3x3, в нашем случае используется поле 5х5 что сути не меняет.

****

**Основной игровой цикл в Main:**

int main()

{

setlocale(LC\_ALL, "ru");

Game g;

initGame(g);

clearScr();

drawGame(g);

do

{

if (g.turn % 2 == 0)

{

// Human

Coord c = getHumanCoord(g);

g.ppField[c.y][c.x] = g.human;

}

else

{

// Comp

Coord c = getAICoord(g);

g.ppField[c.y][c.x] = g.ai;

}

clearScr();// после каждого хода чистим экран

drawGame(g);// отображаем обновленное поле

g.turn++;// увеличиваем очередность хода на 1

g.progress = getWon(g);// кто победил

} while (g.progress == IN\_PROGRESS);

congrats(g);

deinitGame(g);

return 0;

}

Функция Main так же содержит вспомогательные функции очистки экрана:

void clearScr()

{

system("cls");

}

И поздравления победителя:

void \_\_fastcall congrats(const Game& g)

{

if (g.progress == WON\_HUMAN)

{

cout << "Человек выиграл" << endl;

}

else if (g.progress == WON\_AI)

{

cout << "Бот выиграл" << endl;

}

else if (g.progress == DRAW)

{

cout << "Ничья" << endl;

}

}

**Удаление игры:**

void \_\_fastcall deinitGame(Game& g)

{

for (size\_t i = 0; i < g.SIZE; i++)

{

delete[] g.ppField[i];// удаляем строки и клонки матрицы

}

delete[] g.ppField;// чистим память

g.ppField = nullptr;// зануляем указатель

}

ЛИСТИНГ ПРОГРАММЫ

**Содержимое заголовочного файла:**

#pragma once

#include <iostream>

#include <random>

#include <stdlib.h>

#include <chrono>

#include "Header.h"

using namespace std;

// Структуры и перечесления(struct & enum)

// =====================================================================

// =====================================================================

// Перечисление ячеек(элементов) поля

enum Cell : char// Таким образом обЪявляем чаровский тип

{

CROSS = 'X',

ZERO = '0',

EMPTY = '\_'

};

// Структура с координатами

struct Coord

{

size\_t y{ 0 };

size\_t x{ 0 };

};

// Перечисление состояний игры

enum GameProgress

{

IN\_PROGRESS,

WON\_HUMAN,

WON\_AI,

DRAW

};

#pragma pack(push, 1)

struct Game

{

Cell\*\* ppField = nullptr;// указатель на указатель ячеек поля

const uint16\_t SIZE = 5;// Поле 5х5 будет

Cell human;// Кто чем играет

Cell ai;// --//--

size\_t turn = 0;// определение хода.Четный-человек, нечетный-ИИ

GameProgress progress = IN\_PROGRESS;

};

#pragma pack(pop)

//=======================================================================

//=======================================================================

// Функции

void clearScr();

void \_\_fastcall initGame(Game& g);

void \_\_fastcall deinitGame(Game& g);

void \_\_fastcall drawGame(const Game& g);

GameProgress \_\_fastcall getWon(const Game& g);

Coord \_\_fastcall getHumanCoord(const Game& g);

Coord \_\_fastcall getAICoord(const Game& g);

void \_\_fastcall congrats(const Game& g);

int32\_t \_\_fastcall getRundomNum(int32\_t min, int32\_t max);

**Содержимое основного файла:**

#include "Header.h"

// Очистка экрана

void clearScr()

{

system("cls");

}

// C++ный сложный генератор чисел

int32\_t \_\_fastcall getRundomNum(int32\_t min, int32\_t max)

{

const static auto seed = chrono::system\_clock::now().time\_since\_epoch().count();

static mt19937\_64 generator(seed);

uniform\_int\_distribution<int32\_t> dis(min, max);

return dis(generator);

}

//================================================================================

//================================================================================

void \_\_fastcall initGame(Game& g)

{

//////////////////////////////////////////////////////////////////////////////////////////

// Сделал один вложеный цикл с объявлением и инициализацией,вместо двух отдельных

g.ppField = new Cell \* [g.SIZE];

for (size\_t y = 0; y < g.SIZE; y++)// создаем строки матрицы

{

g.ppField[y] = new Cell[g.SIZE];// создаем колонки матрицы

for (size\_t x = 0; x < g.SIZE; x++)

{

g.ppField[y][x] = EMPTY;// инициализируем строки и колонки матрицы

}

}

/////////////////////////////////////////////////////////////////////////////////////////////

if (getRundomNum(0, 1000) > 500)

{

g.human = CROSS;

g.ai = ZERO;

g.turn = 0;

}

else

{

g.human = ZERO;

g.ai = CROSS;

g.turn = 1;

}

}

void \_\_fastcall deinitGame(Game& g)

{

for (size\_t i = 0; i < g.SIZE; i++)

{

delete[] g.ppField[i];// удаляем строки и клонки матрицы

}

delete[] g.ppField;// чистим память

g.ppField = nullptr;// зануляем указатель

}

// Рисуем поле игры

void \_\_fastcall drawGame(const Game& g)

{

// верхние и верхний боковой отступы,нумерация по Х

cout << endl << " ";

for (size\_t x = 0; x < g.SIZE; x++)

{

cout << x + 1 << " ";

}

cout << endl;

// само поле

for (size\_t y = 0; y < g.SIZE; y++)

{

cout << " " << y + 1 << " | ";

for (size\_t x = 0; x < g.SIZE; x++)

{

cout << g.ppField[y][x] << " | ";

}

cout << endl;

}

cout << endl << "Человек: " << g.human << endl << "Бот: " << g.ai << endl;

}

// Кто выиграл

GameProgress \_\_fastcall getWon(const Game& g)

{

// Проверяем поле по строкам(Y)

for (size\_t y = 0; y < g.SIZE; y++)

{

if (g.ppField[y][0] == g.ppField[y][1] && g.ppField[y][0] == g.ppField[y][2] && g.ppField[y][0] == g.ppField[y][3] && g.ppField[y][0] == g.ppField[y][4])

{

if (g.ppField[y][0] == g.human)

{

return WON\_HUMAN;

}

if (g.ppField[y][0] == g.ai)

{

return WON\_AI;

}

}

}

// Проверяем поле по столбцам(X)

for (size\_t x = 0; x < g.SIZE; x++)

{

if (g.ppField[0][x] == g.ppField[1][x] && g.ppField[0][x] == g.ppField[2][x] && g.ppField[0][x] == g.ppField[3][x] && g.ppField[0][x] == g.ppField[4][x])

{

if (g.ppField[0][x] == g.human)

{

return WON\_HUMAN;

}

if (g.ppField[0][x] == g.ai)

{

return WON\_AI;

}

}

}

// 1ая диагональ

if (g.ppField[0][0] == g.ppField[1][1] && g.ppField[0][0] == g.ppField[2][2] && g.ppField[0][0] == g.ppField[3][3] && g.ppField[0][0] == g.ppField[4][4])

{

if (g.ppField[2][2] == g.human)

{

return WON\_HUMAN;

}

if (g.ppField[2][2] == g.ai)

{

return WON\_AI;

}

}

// 2ая диагональ

if (g.ppField[0][4] == g.ppField[1][3] && g.ppField[0][4] == g.ppField[2][2] && g.ppField[0][4] == g.ppField[3][1] && g.ppField[0][4] == g.ppField[4][0])

{

if (g.ppField[2][2] == g.human)

{

return WON\_HUMAN;

}

if (g.ppField[2][2] == g.ai)

{

return WON\_AI;

}

}

// Проверка на ничью

bool draw = true;

for (size\_t y = 0; y < g.SIZE; y++)

{

for (size\_t x = 0; x < g.SIZE; x++)

{

if (g.ppField[y][x] == EMPTY)

{

draw = false;

break;

}

}

if (!draw)

{

break;

}

}

if (draw)

{

return DRAW;

}

return IN\_PROGRESS;

}

Coord \_\_fastcall getHumanCoord(const Game& g)

{

Coord c;

do

{

cout << "Enter X (1..5): ";

cin >> c.x;

cout << "Enter y (1..5): ";

cin >> c.y;

c.x--;

c.y--;

} while (c.x > 4 || c.y > 4 || g.ppField[c.y][c.x] != EMPTY);

return c;

}

Coord \_\_fastcall getAICoord(const Game& g)

{

////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Предвыигрыш и предпроигрыш.Было 2 вложеных цикла,сделал в одном

for (size\_t y = 0; y < g.SIZE; y++)

{

for (size\_t x = 0; x < g.SIZE; x++)

{

if (g.ppField[y][x] == EMPTY)

{

g.ppField[y][x] = g.ai; // Гипотетический ход: комп проверяет ход на выигрышность

if (getWon(g) == WON\_AI) // если после этого хода выиграл

{

g.ppField[y][x] = EMPTY; // возвращает как было

return { y, x }; // и ходит туда

}

g.ppField[y][x] = EMPTY; // в любом случае востанавливаем в пустое(скрываем читерство бота)

g.ppField[y][x] = g.human; // Гипотетический ход от лица человека: комп ходит за человека,проверяет ход на выигрышность

if (getWon(g) == WON\_HUMAN) // если после этого хода,человек выиграл

{

g.ppField[y][x] = EMPTY; // возвращает как было

return { y, x }; // и ходит туда,тем самым мешает человеку выиграть

}

g.ppField[y][x] = EMPTY; // в любом случае востанавливаем в пустое(скрываем читерство бота)

}

}

}

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////

if (g.ppField[2][2] == EMPTY)

{

return { 2, 2 };

}

Coord arr[20];

size\_t num = 0;

if (g.ppField[0][0] == EMPTY) // Ищем все и сохраняем свободные углы

{

arr[num++] = { 0, 0 }; // сначало NUM используется(0), затем увеличивается на 1.

}

if (g.ppField[0][4] == EMPTY)

{

arr[num++] = { 0, 4 }; // NUM = 1(после использования = 2 и тд).

}

if (g.ppField[4][4] == EMPTY)

{

arr[num++] = { 4, 4 };

}

if (g.ppField[4][0] == EMPTY)

{

arr[num++] = { 4, 0 };

}

if (num > 0) // если есть свободные углы

{

const size\_t index = getRundomNum(0, 1000) % num; // случайный индекс свободного угла

return arr[index];

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Не углы.Цикл вместо ифов

for (size\_t y = 0; y < g.SIZE; y++)

{

for (size\_t x = 0; x < g.SIZE; x++)

{

if (g.ppField[y][x] == EMPTY && g.ppField[y][x] != g.ppField[0][0] && g.ppField[y][x] != g.ppField[0][4] && g.ppField[y][x] != g.ppField[4][4] && g.ppField[y][x] != g.ppField[4][0])

{

arr[num++] = { y, x };

}

}

}

if (num > 0) // если есть свободные не углы

{

const size\_t index = getRundomNum(0, 1000) % num; // случайный индекс свободного не угла

return arr[index];

}

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

}

void \_\_fastcall congrats(const Game& g)

{

if (g.progress == WON\_HUMAN)

{

cout << "Человек выиграл" << endl;

}

else if (g.progress == WON\_AI)

{

cout << "Бот выиграл" << endl;

}

else if (g.progress == DRAW)

{

cout << "Ничья" << endl;

}

}

//================================================================================

//================================================================================

**Содержимое Main файла:**

#include "Header.h"

int main()

{

setlocale(LC\_ALL, "ru");

Game g;

initGame(g);

clearScr();

drawGame(g);

do

{

if (g.turn % 2 == 0)

{

// Human

Coord c = getHumanCoord(g);

g.ppField[c.y][c.x] = g.human;

}

else

{

// Comp

Coord c = getAICoord(g);

g.ppField[c.y][c.x] = g.ai;

}

clearScr();// после каждого хода чистим экран

drawGame(g);// отображаем обновленное поле

g.turn++;// увеличиваем очередность хода на 1

g.progress = getWon(g);// кто победил

} while (g.progress == IN\_PROGRESS);

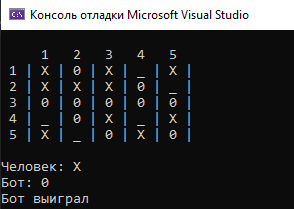
congrats(g);

deinitGame(g);

return 0;

}

**Результат выполнения программы:**

****

ЗАКЛЮЧЕНИЕ

Как мы можем заметить, практика программирования состоит не только и не столько из знаний о синтаксисе того или иного языка программирования, но из решений задач, составления алгоритмов действий и применения изученных конструкций в уместном контексте. При решении практических задач важно помнить две вещи: - что компьютер не имеет никакого жизненного опыта, поэтому не сможет выполнить действие, которое “ну тут и так понятно, что надо”; - что работа компьютера не является магией, и компьютер всегда в точности делает именно то, что написал в своём коде программист. Ключ к решению любой сложной задачи - это её декомпозиция до понятных конструкций. В нашем случае, до понятных компьютеру конструкций.

ЛИТЕРАТУРА

Пол Дейтел, Харви Дейтел. “Как программировать га С++” – Бином 2021.