Федеральное агентство связи  
Федеральное государственное бюджетное  
образовательное учреждение высшего образования  
 «Сибирский государственный университет  
телекоммуникаций и информатики»

Кафедра вычислительных систем

Курсовая работа

по дисциплине «Технологии разработки программного обеспечения»

на тему “Игра - Висилица”.

Выполнил:

ст. гр. ИС-242

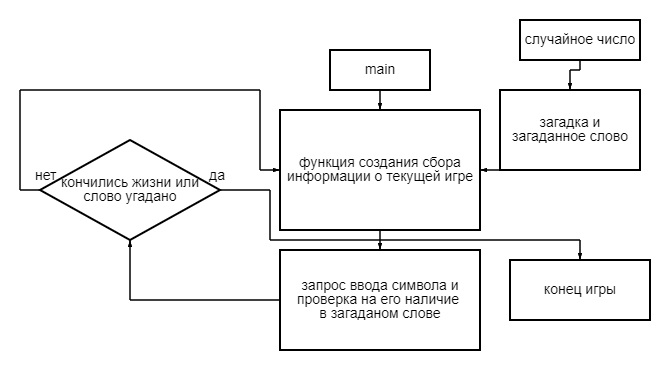
Повышев Д. В.

Проверил:

ст. преподаватель Токмашева Е. И.

# Введение и постановка задачи

Главная цель состоит в том, чтобы реализовать игру под названием “Висилица”. Наша идея её реализации состоит в следующем плане:



То есть, необходимо реализовать функции:

1. Заполнения структуры, включающей в себя, кол-во жизней, загаданное слово, угадываемое слово, номер загадки.
2. Генерации случайных чисел.
3. Загадки и загаданные слова.
4. Проверка на наличие символа в загадке.
5. Проверка условия количества жизней или угаданного слова
6. Соответствующий конец игры.

# 

# Техническое задание

Цель игры:

Угадать слово за определённое количество попыток

Требования к функциональным характеристикам:

- Управление с помощью клавиатуры

Возможный функционал программы:

- Ввод символов с клавиатуры

Логика программы:

После запуска появляется интерфейс для взаимодействия с пользователем.

Пользователю предоставляется выбор:

Начать новую игру или выйти.

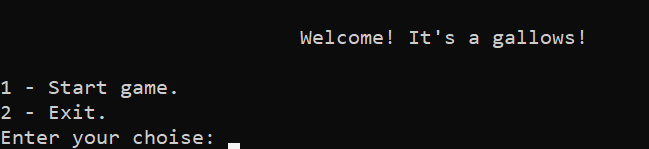
Пользователь видит загадку, ответом на которую является оаределённое слово, которое нужно отгадать.

Процесс разгадывания слово происходит следующим образом:

Пользователь вводит букву. Если такая имеется в загаданном слове, игра продолжается. Если нет - у игрока отнимается одна жизнь.

# Описание выполненного проекта

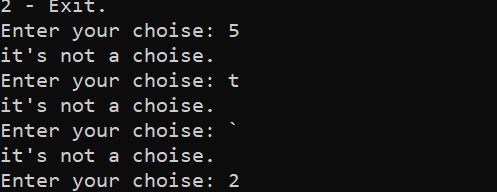
При запуске исполняемого файла, на экран выводится сообщение и предоставляется выбор:



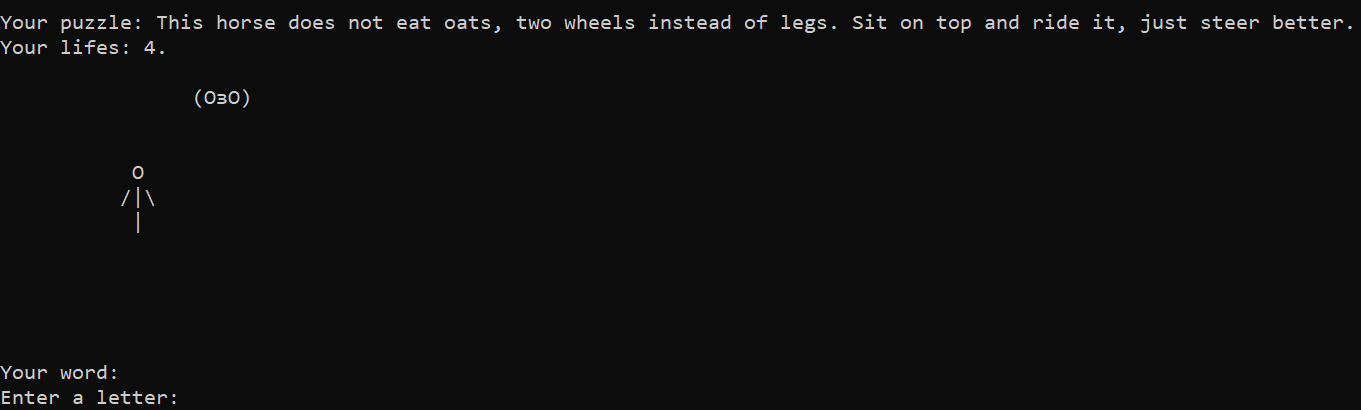
Если пользователь выберет второй вариант – работа исполняемого файла завершится.

Выбрав первое, пользователь запустит процесс сбора информации о текущей игре и она начнётся.

При этом, если ввести символ, не являющийся двойкой или единицей, программа попросит ввести одну из перечисленных команд снова:



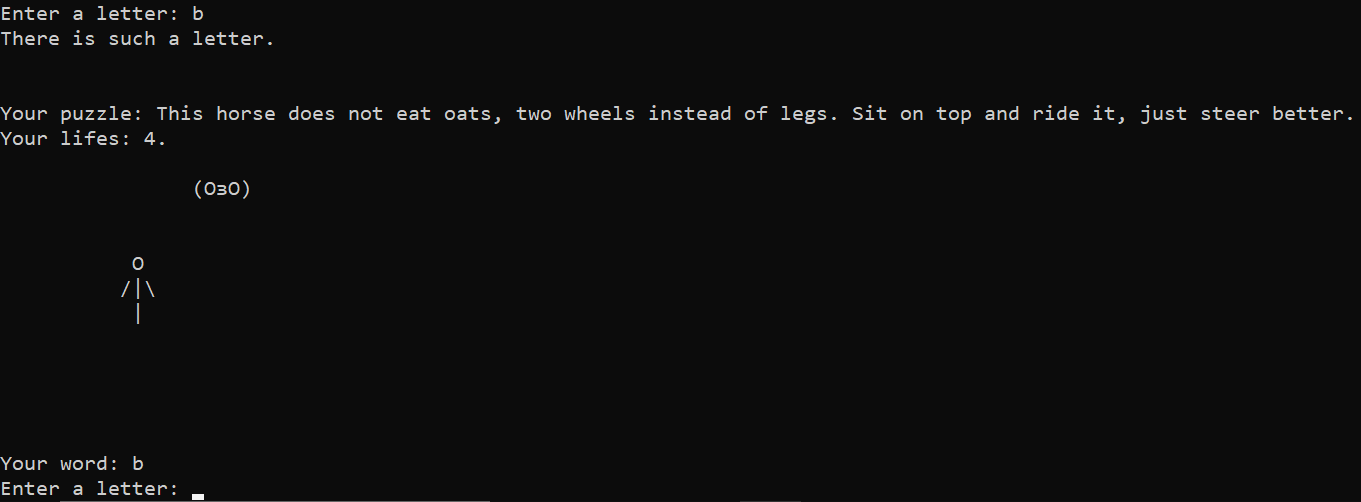
После того, как пользователь вводит команду для начала игры, выводится следующее:



Сверху – загадка, которая позволяет угадать загаданное слово, чуть ниже – оставшиеся жизни, также есть интерфейс, представляющий из себя человечка и смайлика сверху (для наглядности).

Ниже поле для угаданного слова. Там появляются новые буквы в зависимости от тех, что мы вводим уже в поле ниже.

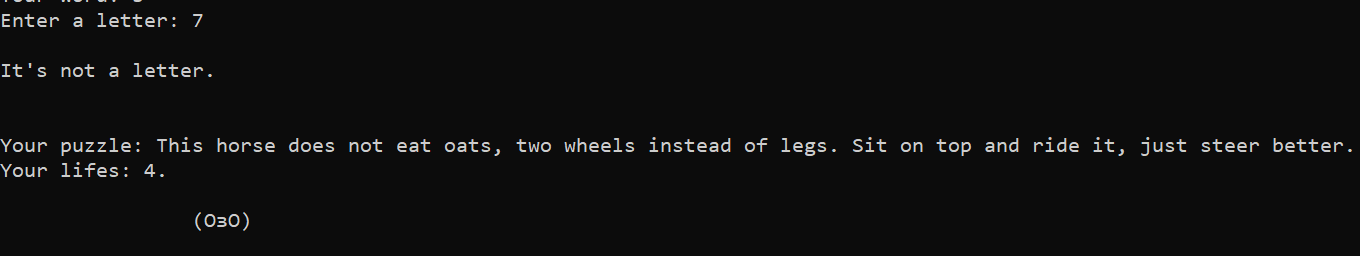
Если буква угадана правильно – она появляется в поле “Your word” в соответствии с положением в реальном слове.



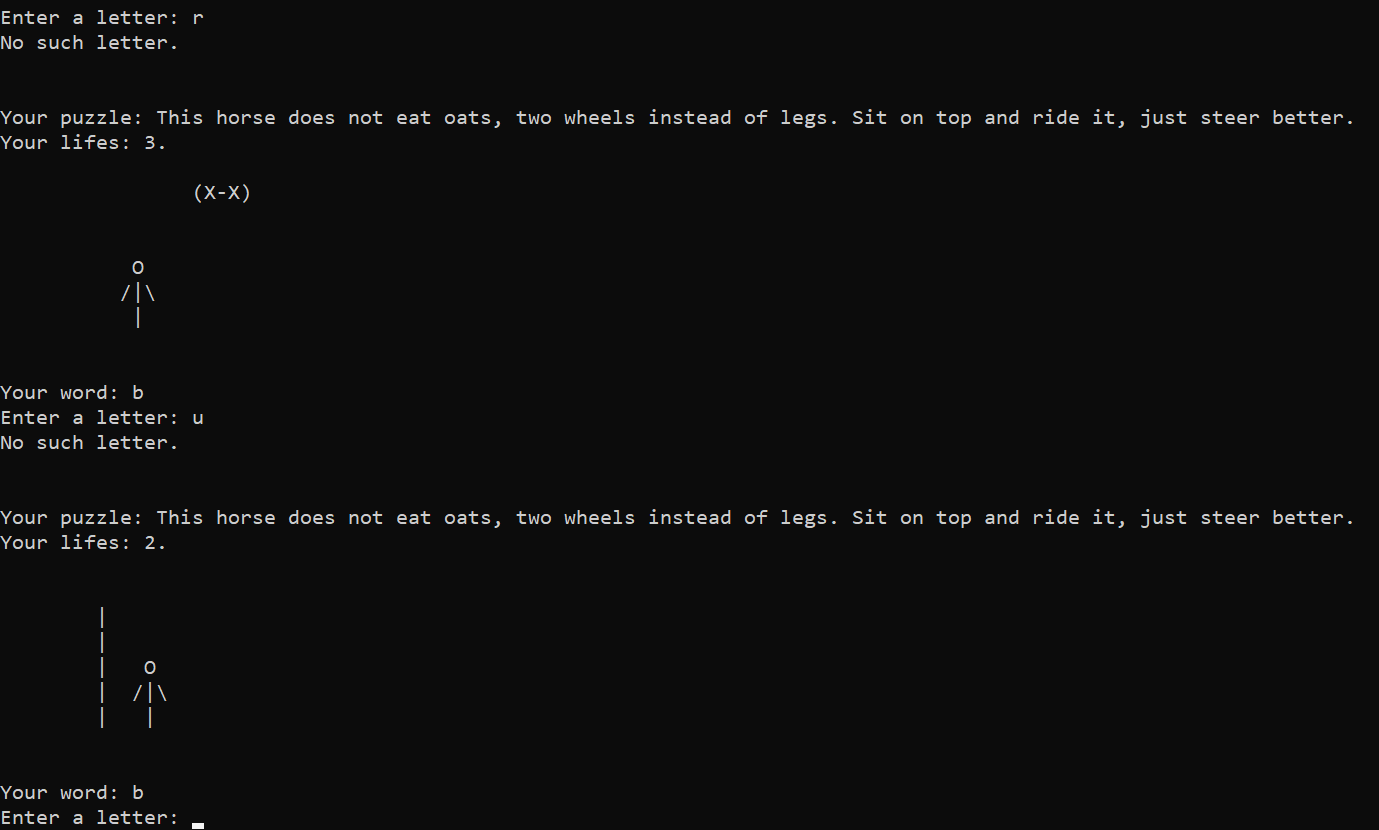
Пользователь догадался, что загаданное слово “bike” и ввёл первую букву.

Она оказалась правильной и появилась в поле угадываемого слова.

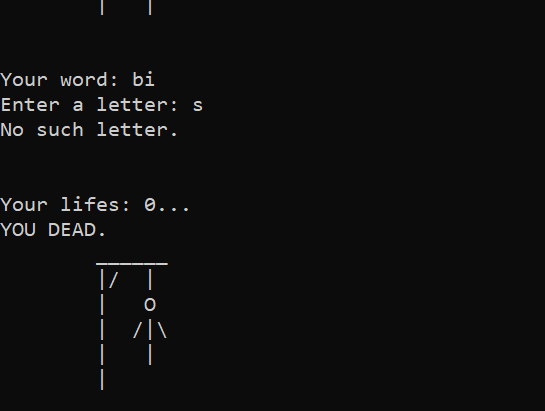
Программе не важно, какой у буквы регистр. Если же ввести не что-либо кроме букв английского алфавита, программа попросит повторить попытку:



Далее пользователь допустил несколько ошибок и получил следующее:

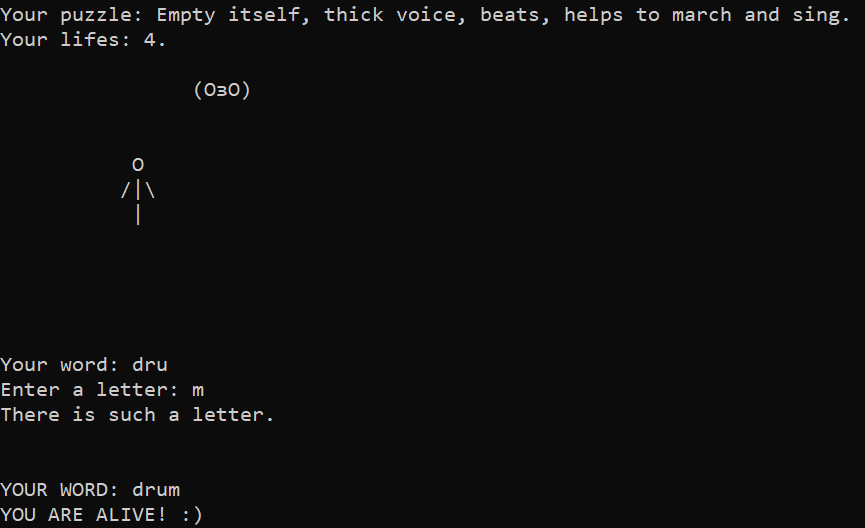


Загаданное слово было не “bike”. Пользователь проиграл:



Так выглядит интерфейс в случае проигрыша.

Если же пользователю удастся угадать слово, получится:



На этом функционал программы окончен.

# Личный вклад в проект

Мной в данном проекте были реализованы функции работы со структурой, проверки буквы на наличие в слове и другие функции, исключающие баги во время работы игры:

Strlower, errors, custom\_block, examination\_letter, is\_it\_letter.

Также были реализованы:

Часть Makefile, Github Actions, проверяющие файлы проекта на форматирование текста и работоспособность Makefile.

Помимо этого, было множество моментов, которые решались и реализовывались совместно за счёт github issues и живого обсуждения проекта.

# Приложение. Текст программы

Main.c:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | #include <ctype.h>  #include <enterface.h>  #include <puzzles.h>  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <time.h>  #include <user.h>  **int** **main**()  {  printf("**\n\n\t\t\t** Welcome! It's a gallows!**\n\n**");  printf("1 - Start game.**\n**");  printf("2 - Exit.**\n**");  **int** choise = **5**;  **while** (choise != **1** && choise != **2**)  {  printf("Enter your choise: ");  **char** c;  scanf("%d", &choise);  **while** ((c = getchar() != '\n' && c != EOF)); //Классная штука, чистит буфер  **if** (choise != **1** && choise != **2**) {  printf("This is the wrong choice.**\n**");  }  }  printf("**\n\n\n**");  **if** (choise == **1**) {  My\_game\* we\_have = get\_data\_for\_start\_game();  custom\_block(we\_have);  }  **return** **0**;  } |

Enterface.c:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82 | #include <ctype.h>  #include <enterface.h>  #include <puzzles.h>  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <time.h>  **void** **print\_duck\_guess**(My\_game\* we\_have)  {  printf("**\n\n**Your word: ");  **for** (**int** i = **0**; i < **sizeof**(we\_have->duck\_guess - **2**); i++) {  **if** (we\_have->duck\_guess[i] == **0**) {  printf(" ");  } **else** {  printf("%c", we\_have->duck\_guess[i]);  }  }  printf("**\n**");  }  **int** **enteface**(My\_game\* we\_have)  {  **switch** (we\_have->life) {  **case** **0**:  printf("Your lifes: 0...**\n**");  printf("YOU DEAD.**\n**");  printf("**\t**\_\_\_\_\_\_**\n**");  printf("**\t**|/ |**\n**");  printf("**\t**| O**\n**");  printf("**\t**| /|**\\\n**");  printf("**\t**| |**\n**");  printf("**\t**|**\n**");  printf("**\n\n\n**");  **break**;  **case** **1**:  printf("Your lifes: 1.**\n**");  printf("**\n**");  printf("**\t**\_\_\_\_\_\_**\n**");  printf("**\t**|/ **\n**");  printf("**\t**| **\n**");  printf("**\t**| O**\n**");  printf("**\t**| /|**\\\n**");  printf("**\t**| |**\n**");  **break**;  **case** **2**:  printf("Your lifes: 2.**\n**");  printf("**\n**");  printf("**\t\n**");  printf("**\t**| **\n**");  printf("**\t**| **\n**");  printf("**\t**| O**\n**");  printf("**\t**| /|**\\\n**");  printf("**\t**| |**\n**");  **break**;  **case** **3**:  printf("Your lifes: 3.**\n**");  printf("**\n**");  printf("**\t** (X-X)**\n**");  printf("**\t** **\n**");  printf("**\t** **\n**");  printf("**\t** O**\n**");  printf("**\t** /|**\\\n**");  printf("**\t** |**\n**");  **break**;  **case** **4**:  printf("Your lifes: 4.**\n**");  printf("**\n**");  printf("**\t** (OзO)**\n**");  printf("**\t** **\n**");  printf("**\t** **\n**");  printf("**\t** O**\n**");  printf("**\t** /|**\\\n**");  printf("**\t** |**\n**");  printf("**\n\n\n**");  **break**;  **default:**  printf("Are you cheater?**\n**");  **return** -**1**;  }  **return** **0**;  } |

Puzzles.c:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247 | #include <ctype.h>  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <time.h>  #include <user.h>  **char**\* **puzzles**(**int** number, **int** print)  {  **char**\* duck;  duck = malloc(**20**);  **switch** (number) {  **case** **1**:  **if** (print == **1**) {  printf("Under the mountain is a pony, in the mountain - a wooden "  "thing.**\n**");  } **else** {  duck = "sled";  }  **break**;  **case** **2**:  **if** (print == **1**) {  printf("Flowers fall from the sky onto trees and bushes. White, "  "fluffy, but not fragrant.**\\**n");  } **else** {  duck = "snow";  }  **break**;  **case** **3**:  **if** (print == **1**) {  printf("A red maiden sits in a dungeon and her braid is on the "  "street.**\n**");  } **else** {  duck = "carrot";  }  **break**;  **case** **4**:  **if** (print == **1**) {  printf("In winter - a star, in spring - water.**\n**");  } **else** {  duck = "snowflake";  }  **break**;  **case** **5**:  **if** (print == **1**) {  printf("Who is angry and hungry in cold winter?**\n**");  } **else** {  duck = "wolf";  }  **break**;  **case** **6**:  **if** (print == **1**) {  printf("It doesn't bark, doesn't bite, but doesn't let you into "  "the house.**\n**");  } **else** {  duck = "lock";  }  **break**;  **case** **7**:  **if** (print == **1**) {  printf("A hundred clothes with no fasteners.**\n**");  } **else** {  duck = "cabbage";  }  **break**;  **case** **8**:  **if** (print == **1**) {  printf("An old man sits dressed in a fur coat, whoever undresses "  "him sheds tears.**\n**");  } **else** {  duck = "onion";  }  **break**;  **case** **9**:  **if** (print == **1**) {  printf("Little white pigs lay on the ridges.**\n**");  } **else** {  duck = "zucchini";  }  **break**;  **case** **10**:  **if** (print == **1**) {  printf("It shines, it sparkles, it doesn't flatter anyone.**\n**");  } **else** {  duck = "mirror";  }  **break**;  **case** **11**:  **if** (print == **1**) {  printf("A poor fellow hides in the board - only his cap is "  "visible.**\n**");  } **else** {  duck = "nail";  }  **break**;  **case** **12**:  **if** (print == **1**) {  printf("In winter and in summer, it is the same color.**\n**");  } **else** {  duck = "tree";  }  **break**;  **case** **13**:  **if** (print == **1**) {  printf("At first, there is gloss, then - a crack!**\n**");  } **else** {  duck = "thunderstorm";  }  **break**;  **case** **14**:  **if** (print == **1**) {  printf("A red-haired rogue, cunning and skillful, came into the "  "barn and recounted the hens.**\n**");  } **else** {  duck = "fox";  }  **break**;  **case** **15**:  **if** (print == **1**) {  printf("He walks importantly across the puddle, comes out of the "  "water dry, wears red boots, gives soft featherbeds.**\n**");  } **else** {  duck = "goose";  }  **break**;  **case** **16**:  **if** (print == **1**) {  printf("Striped balls from the garden have arrived here.**\n**");  } **else** {  duck = "watermelon";  }  **break**;  **case** **17**:  **if** (print == **1**) {  printf("White peas on a green leg.**\n**");  } **else** {  duck = "lily";  }  **break**;  **case** **18**:  **if** (print == **1**) {  printf("Teeth in the mouth in three rows. It's a whole disaster. "  "This predator is famous for being a sea bandit.**\n**");  } **else** {  duck = "shark";  }  **break**;  **case** **19**:  **if** (print == **1**) {  printf("We are forest inhabitants, wise builders. We all together "  "build a house for ourselves under the spruce from "  "needles.**\n**");  } **else** {  duck = "ants";  }  **break**;  **case** **20**:  **if** (print == **1**) {  printf("A log floats down the river. Oh, and it's so wicked!**\n**");  } **else** {  duck = "crocodile";  }  **break**;  **case** **21**:  **if** (print == **1**) {  printf("The living castle growls, lies across the door. Two medals "  "on his chest. Better not to go in!**\n**");  } **else** {  duck = "dog";  }  **break**;  **case** **22**:  **if** (print == **1**) {  printf("I wake everyone up on time, even though I don't wind the "  "clocks.**\n**");  } **else** {  duck = "rooster";  }  **break**;  **case** **23**:  **if** (print == **1**) {  printf("He knocks all the time, pecks trees. But he doesn't "  "cripple them, only heals them.**\n**");  } **else** {  duck = "woodpecker";  }  **break**;  **case** **24**:  **if** (print == **1**) {  printf("Empty itself, thick voice, beats, helps to march and "  "sing.**\n**");  } **else** {  duck = "drum";  }  **break**;  **case** **25**:  **if** (print == **1**) {  printf("It will fall - it will bounce, you will hit - it will not "  "cry.**\n**");  } **else** {  duck = "ball";  }  **break**;  **case** **26**:  **if** (print == **1**) {  printf("Oh, do not touch me: I will burn you even without fire!**\n**");  } **else** {  duck = "nettle";  }  **break**;  **case** **27**:  **if** (print == **1**) {  printf("This horse does not eat oats, two wheels instead of legs. "  "Sit on top and ride it, just steer better.**\n**");  } **else** {  duck = "bicycle";  }  **break**;  **case** **28**:  **if** (print == **1**) {  printf("Then back, then forth, wanders, floats the ship. If you "  "stop it - beware! It will poke holes in the sea!**\n**");  } **else** {  duck = "iron";  }  **break**;  **case** **29**:  **if** (print == **1**) {  printf("Admire, look - the North Pole inside! There shines snow "  "and ice, there winter itself lives.**\n**");  } **else** {  duck = "refrigerator";  }  **break**;  **case** **30**:  **if** (print == **1**) {  printf("I can take you underground any time of the year and in any "  "weather, and quickly take you anywhere in an hour.**\n**");  } **else** {  duck = "subway";  }  **break**;  **default:**  errors(**1**);  }  **return** duck;  } |

User.c:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105 | #include <ctype.h>  #include <enterface.h>  #include <puzzles.h>  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <time.h>  #include <user.h>  **int** **errors**(**int** error\_num)  {  **switch** (error\_num) {  **case** **1**:  printf("Ошибка в номере загадки.**\n**");  **break**;  **default:**  printf("Хэй, такакя ошибка вообще бывает?**\n**");  **break**;  }  **return** -**1**;  }  **void** **strlower**(**char**\* string)  {  **for** (**int** i = **0**; i < strlen(string); i++) {  string[i] = tolower(string[i]);  }  }  **int** **generate\_number**()  {  srand(time(NULL));  **int** random\_value = **1** + rand() % **30**;  **return** random\_value;  }  My\_game\* **get\_data\_for\_start\_game**()  {  My\_game\* we\_have = malloc(**sizeof**(My\_game));  we\_have->puzzle\_num = generate\_number();  we\_have->duck = puzzles(we\_have->puzzle\_num, **0**);  we\_have->life = **4**;  we\_have->duck\_guess = calloc(**1**, **sizeof**(we\_have->duck));  **return** we\_have;  }  **void** **examination\_letter**(My\_game\* we\_have, **char** next\_letter)  {  **int** count\_letter = **0**;  **for** (**int** i = **0**; i < **sizeof**(we\_have->duck); i++) {  **if** (we\_have->duck[i] == next\_letter) {  we\_have->duck\_guess[i] = next\_letter;  count\_letter++;  }  }  **if** (count\_letter >= **1**) {  printf("There is such a letter.**\n\n\n**");  } **else** {  printf("No such letter.**\n\n\n**");  we\_have->life--;  }  }  **int** **is\_it\_letter**(**char** next\_letter)  {  **if** ((next\_letter >= **65** && next\_letter <= **90**)  || (next\_letter >= **97** && next\_letter <= **122**)) {  **return** **1**;  } **else** {  **return** **0**;  }  }  **void** **custom\_block**(My\_game\* we\_have)  {  **char** c;  **while** (we\_have->life > **0**  && strcmp(we\_have->duck\_guess, we\_have->duck) != **0**) {  **int** is\_it\_letter\_exam = **0**;  **char** next\_letter;  **while** (is\_it\_letter\_exam == **0**) {  printf("Your puzzle: ");  puzzles(we\_have->puzzle\_num, **1**);  enteface(we\_have);  print\_duck\_guess(we\_have);  printf("Enter a letter: ");  next\_letter = getchar();  **while** ((c = getchar() != '\n' && c != EOF))  ; //Классная штука, чистит буфер  is\_it\_letter\_exam = is\_it\_letter(next\_letter);  **if** (is\_it\_letter\_exam == **0**) {  printf("**\n**It's not a letter.**\n\n\n**");  }  }  strlower(&next\_letter);  examination\_letter(we\_have, next\_letter);  }  **if** (we\_have->life <= **0**) {  enteface(we\_have);  }  **if** (strcmp(we\_have->duck\_guess, we\_have->duck) == **0** && we\_have->life > **0**) {  printf("YOUR WORD: %s**\n**", we\_have->duck);  printf("YOU ARE ALIVE! :)**\n**");  }  } |

Enterface.h:

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

#ifndef ENTERFACE\_H

#define ENTERFACE\_H

#include <stdio.h>

#include <time.h>

#include <string.h>

#include <stdlib.h>

#include <ctype.h>

#include <puzzles.h>

#include <enterface.h>

**void** **print\_duck\_guess**(My\_game\* we\_have);

**int** **enteface**(My\_game\* we\_have);

#endif //ENTERFACE\_H

Puzzles.h:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | #ifndef PUZZLES\_H  #define PUZZLES\_H  #include <stdio.h>  #include <time.h>  #include <string.h>  #include <stdlib.h>  #include <ctype.h>  **char** \***puzzles**(**int** number, **int** print);  **typedef** **struct**  {  **int** life;  **int** puzzle\_num;  **char** \*duck;  **char** \*duck\_guess;  } My\_game;  #endif //PUZZLES\_H |

User.h:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | #ifndef USER\_H  #define USER\_H  #include <stdio.h>  #include <time.h>  #include <string.h>  #include <stdlib.h>  #include <ctype.h>  #include <puzzles.h>  #include <enterface.h>  #include <user.h>  **int** **errors**(**int** error\_num);  **void** **strlower**(**char**\* string);  **int** **generate\_number**();  My\_game\* **get\_data\_for\_start\_game**();  **void** **examination\_letter**(My\_game\* we\_have, **char** next\_letter);  **int** **is\_it\_letter**(**char** next\_letter);  **void** **custom\_block**(My\_game\* we\_have);  #endif //USER\_H |

Makefile:

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

TARGET = ./bin/MyProject

CC = gcc

CFLAGS = -Wall

PATH\_INCLUDE = -Isrc

FORMAT\_FILES = clang-format -i --verbose

PREF\_SRC = ./src/

PREF\_OBJ = ./obj/src/

SRC = **$(**wildcard **$(**PREF\_SRC**)**\*.c**)**

OBJ = **$(**patsubst **$(**PREF\_SRC**)**%.c, **$(**PREF\_OBJ**)**%.o, **$(**SRC**))**

**all**: **$(TARGET)**

**$(**FORMAT\_FILES**)** **$(**SRC**)**

**$(**PREF\_OBJ**)**%.o : **$(**PREF\_SRC**)**%.c

**$(**CC**)** -c **$(**CFLAGS**)** **$(**PATH\_INCLUDE**)** -o $@ $<

**$(**TARGET**)** : **$(**OBJ**)** **$(**FORMAT**)**

**$(**CC**)** **$(**CFLAGS**)** -o **$(**TARGET**)** **$(**OBJ**)**

clean :

rm **$(**TARGET**)** **$(**PREF\_OBJ**)**\*.o

TESTS\_SRC = **$(**wildcard ./tests/\*.c**)**

tests :

**$(**CC**)** **$(**CFLAGS**)** **$(**PATH\_INCLUDE**)** -o ./bin/test-puzzles ./test/test-puzzles.c ./src/puzzles.c ./src/user.c ./src/enterface.c

**$(**CC**)** **$(**CFLAGS**)** **$(**PATH\_INCLUDE**)** -o ./bin/test-enterface ./test/test-enterface.c ./src/puzzles.c ./src/user.c ./src/enterface.c

**$(**CC**)** **$(**CFLAGS**)** **$(**PATH\_INCLUDE**)** -o ./bin/test-user ./test/test-user.c ./src/puzzles.c ./src/user.c ./src/enterface.c

Test-enterface.c:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77 | #include <stdio.h>  #include <assert.h>  #include <string.h>  #include <stdlib.h>  #include <ctype.h>  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <time.h>  #include <user.h>  #include <puzzles.h>  **void** **test\_enteface**()  {  My\_game game;  // Тест 1: проверка вывода для 0 жизней  game.life = **0**;  assert(enteface(&game) == **0**);  // Тест 2: проверка вывода для 1 жизни  game.life = **1**;  assert(enteface(&game) == **0**);  // Тест 3: проверка вывода для 2 жизней  game.life = **2**;  assert(enteface(&game) == **0**);  // Тест 4: проверка вывода для 3 жизней  game.life = **3**;  assert(enteface(&game) == **0**);  // Тест 5: проверка вывода для 4 жизней  game.life = **4**;  assert(enteface(&game) == **0**);  // Тест 6: проверка вывода для некорректного числа жизней  game.life = **5**;  assert(enteface(&game) == -**1**);  // Тест 7: проверка вывода для отрицательного числа жизней  game.life = -**1**;  assert(enteface(&game) == -**1**);  }  **void** **test\_print\_duck\_guess**()  {  My\_game game;  // Тест 1: проверка вывода пустой строки  strcpy(game.duck\_guess, "");  print\_duck\_guess(&game);  // Тест 2: проверка вывода строки "hello"  strcpy(game.duck\_guess, "hello");  print\_duck\_guess(&game);  // Тест 3: проверка вывода строки с пробелами  strcpy(game.duck\_guess, " ");  print\_duck\_guess(&game);  // Тест 4: проверка вывода строки с символами и пробелами  strcpy(game.duck\_guess, "a b c");  print\_duck\_guess(&game);  // Тест 5: проверка вывода строки с символами, цифрами и специальными символами  strcpy(game.duck\_guess, "Hello123!@#");  print\_duck\_guess(&game);  }  **int** **main**(**void**)  {  test\_enteface();  test\_print\_duck\_guess();  **return** **0**;  } |

Test-puzzles.c:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175 | #include <stdio.h>  #include <assert.h>  #include <string.h>  #include <stdlib.h>  #include <ctype.h>  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <time.h>  #include <user.h>  #include <puzzles.h>  **void** **test\_puzzles**()  {  **char**\* result;  // Тест 1  result = puzzles(**1**, **0**);  printf("Puzzle 1: %s**\n**", result);  assert(strcmp(result, "sled") == **0**);  // Тест 2  result = puzzles(**2**, **0**);  printf("Puzzle 2: %s**\n**", result);  assert(strcmp(result, "snow") == **0**);  // Тест 3  result = puzzles(**3**, **0**);  printf("Puzzle 3: %s**\n**", result);  assert(strcmp(result, "carrot") == **0**);  // Тест 4  result = puzzles(**4**, **0**);  printf("Puzzle 4: %s**\n**", result);  assert(strcmp(result, "snowflake") == **0**);  // Тест 5  result = puzzles(**5**, **0**);  printf("Puzzle 5: %s**\n**", result);  assert(strcmp(result, "wolf") == **0**);  // Тест 6  result = puzzles(**6**, **0**);  printf("Puzzle 6: %s**\n**", result);  assert(strcmp(result, "lock") == **0**);  // Тест 7  result = puzzles(**7**, **0**);  printf("Puzzle 7: %s**\n**", result);  assert(strcmp(result, "cabbage") == **0**);  // Тест 8  result = puzzles(**8**, **0**);  printf("Puzzle 8: %s**\n**", result);  assert(strcmp(result, "onion") == **0**);  // Тест 9  result = puzzles(**9**, **0**);  printf("Puzzle 9: %s**\n**", result);  assert(strcmp(result, "zucchini") == **0**);  // Тест 10  result = puzzles(**10**, **0**);  printf("Puzzle 10: %s**\n**", result);  assert(strcmp(result, "mirror") == **0**);  // Тест 11  result = puzzles(**11**, **0**);  printf("Puzzle 11: %s**\n**", result);  assert(strcmp(result, "nail") == **0**);  // Тест 12  result = puzzles(**12**, **0**);  printf("Puzzle 12: %s**\n**", result);  assert(strcmp(result, "tree") == **0**);  // Тест 13  result = puzzles(**13**, **0**);  printf("Puzzle 13: %s**\n**", result);  assert(strcmp(result, "thunderstorm") == **0**);  // Тест 14  result = puzzles(**14**, **0**);  printf("Puzzle 14: %s**\n**", result);  assert(strcmp(result, "fox") == **0**);  // Тест 15  result = puzzles(**15**, **0**);  printf("Puzzle 15: %s**\n**", result);  assert(strcmp(result, "goose") == **0**);  // Тест 16  result = puzzles(**16**, **0**);  printf("Puzzle 16: %s**\n**", result);  assert(strcmp(result, "watermelon") == **0**);  // Тест 17  result = puzzles(**17**, **0**);  printf("Puzzle 17: %s**\n**", result);  assert(strcmp(result, "lily") == **0**);  // Тест 18  result = puzzles(**18**, **0**);  printf("Puzzle 18: %s**\n**", result);  assert(strcmp(result, "shark") == **0**);  // Тест 19  result = puzzles(**19**, **0**);  printf("Puzzle 19: %s**\n**", result);  assert(strcmp(result, "ants") == **0**);  // Тест 20  result = puzzles(**20**, **0**);  printf("Puzzle 20: %s**\n**", result);  assert(strcmp(result, "crocodile") == **0**);  // Тест 21  result = puzzles(**21**, **0**);  printf("Puzzle 21: %s**\n**", result);  assert(strcmp(result, "dog") == **0**);  // Тест 22  result = puzzles(**22**, **0**);  printf("Puzzle 22: %s**\n**", result);  assert(strcmp(result, "rooster") == **0**);  // Тест 23  result = puzzles(**23**, **0**);  printf("Puzzle 23: %s**\n**", result);  assert(strcmp(result, "woodpecker") == **0**);  // Тест 24  result = puzzles(**24**, **0**);  printf("Puzzle 24: %s**\n**", result);  assert(strcmp(result, "drum") == **0**);  // Тест 25  result = puzzles(**25**, **0**);  printf("Puzzle 25: %s**\n**", result);  assert(strcmp(result, "ball") == **0**);  // Тест 26  result = puzzles(**26**, **0**);  printf("Puzzle 26: %s**\n**", result);  assert(strcmp(result, "nettle") == **0**);  // Тест 27  result = puzzles(**27**, **0**);  printf("Puzzle 27: %s**\n**", result);  assert(strcmp(result, "bicycle") == **0**);  // Тест 28  result = puzzles(**28**, **0**);  printf("Puzzle 28: %s**\n**", result);  assert(strcmp(result, "iron") == **0**);  // Тест 29  result = puzzles(**29**, **0**);  printf("Puzzle 29: %s**\n**", result);  assert(strcmp(result, "refrigerator") == **0**);  // Тест 30  result = puzzles(**30**, **0**);  printf("Puzzle 30: %s**\n**", result);  assert(strcmp(result, "subway") == **0**);  }  **int** **main**()  {  test\_puzzles();  printf("All tests passed!**\n**");  **return** **0**;  }  Test-user.c:   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82 | #include <stdio.h>  #include <assert.h>  #include <string.h>  #include <stdlib.h>  #include <ctype.h>  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <time.h>  #include <user.h>  #include <puzzles.h>  // Тестируем функцию generate\_number()  **void** **test\_generate\_number**()  {  **int** number = generate\_number();  printf("Generated number: %d**\n**", number);  // Проверяем, что сгенерированное число находится в ожидаемом диапазоне  assert(number >= **1** && number <= **30**);  }  // Тестируем функцию strlower()  **void** **test\_strlower**()  {  **char** string[] = "HeLLo";  strlower(string);  printf("Lowercased string: %s**\n**", string);  // Проверяем, что все символы строки стали в нижнем регистре  assert(string[**0**] == 'h' && string[**1**] == 'e' && string[**2**] == 'l' && string[**3**] == 'l' && string[**4**] == 'o');  }  // Тестируем функцию examination\_letter()  **void** **test\_examination\_letter**()  {  My\_game game;  game.duck = "hello";  game.duck\_guess = calloc(strlen(game.duck) + **1**, **sizeof**(**char**));  game.life = **4**;  examination\_letter(&game, 'l');  printf("Duck guess: %s**\n**", game.duck\_guess);  printf("Remaining life: %d**\n**", game.life);  // Проверяем, что буква 'l' была найдена в слове и вставлена в массив duck\_guess  assert(game.duck\_guess[**2**] == 'l');  // Проверяем, что количество оставшихся жизней уменьшилось  assert(game.life == **4**);  examination\_letter(&game, 'z');  printf("Duck guess: %s**\n**", game.duck\_guess);  printf("Remaining life: %d**\n**", game.life);  // Проверяем, что буква 'z' не была найдена в слове и количество оставшихся жизней уменьшилось  assert(game.duck\_guess[**2**] == 'l');  assert(game.life == **3**);  free(game.duck\_guess);  }  // Тестируем функцию is\_it\_letter()  **void** **test\_is\_it\_letter**()  {  // Проверяем, что функция правильно определяет буквы  assert(is\_it\_letter('A') == **1**);  assert(is\_it\_letter('z') == **1**);  // Проверяем, что функция правильно определяет не-буквы  assert(is\_it\_letter('1') == **0**);  assert(is\_it\_letter('$') == **0**);  }  **int** **main**()  {  test\_generate\_number();  test\_strlower();  test\_examination\_letter();  test\_is\_it\_letter();  printf("All tests passed!**\n**");  **return** **0**;  } |   Workfolows.yml:   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | **name:** Build  **on:** push  **jobs:**  **build:**  runs-on: ubuntu-**20.04**  **steps:**  - uses: actions/checkout@v2  - name: Check codestyle  **run:** git ls-files \*.c | xargs clang-format -i --verbose && git diff --exit-code  - name: Build application  **run:** make | |