

## ТЕМА: ВВЕДЕНИЕ В ЯЗЫК ПРОГРАММИРОВАНИЯ «C++»

### Домашнее задание 1

#### ЗАДАНИЕ 1

Написать программу, которая выводит на экран линию заданным символом, вертикальную или горизонтальную, причем линия может выводиться быстро, нормально и медленно. Общение с пользователем организовать через меню.

##### Подсказка 1

Если у Вас возникли сложности с выполнением данного задания, нажмите кнопку «Подсказка». С полным решением задания вы сможете ознакомиться, нажав на кнопку «Решение»

##### Решение 1

#### ЗАДАНИЕ 2

Написать игру «Кубики». Пользователь и компьютер по очереди бросают 2 кубика. Победитель — тот, у кого по результатам трех бросков сумма больше. Предусмотреть красивый интерфейс игры.

##### Подсказка 2

Если у Вас возникли сложности с выполнением данного задания, нажмите кнопку «Подсказка». С полным решением задания вы сможете ознакомиться, нажав на кнопку «Решение»

##### Решение 2

#### ЗАДАНИЕ 3

Написать игру «GUESS MY NUMBER». Пользователь угадывает число, которое «загадал» компьютер. Возможны 2 уровня сложности: на первом уровне число от 1 до 10, на втором от 10 до 100. В начале игры пользователь выбирает уровень.

В начале каждого уровня игроку даются «жизни» — 50% от длины диапазона угадывания на первом уровне и 25% — на втором (пример приведен ниже). За каждую ошибку игрок штрафуются — минус одна жизнь. Предусмотреть в игре возможность подсказки: «загаданное число больше — меньше твоего». Стоимость подсказки — одна жизнь (пользователь может принять подсказку в случае ошибки или отказаться от нее). Уровень продолжается, пока игрок жив или не угадает (состояние игрока «жив» — «убит» отображается рисунком в консоли).

Очки после уровня — количество оставшихся жизней, умноженные на 5 для первого уровня и на 10 для второго. Результат игры, количество очков или «Проигрыш», вывести в консоль.

**Подсказка 3**

Если у Вас возникли сложности с выполнением данного задания, нажмите кнопку «Подсказка». С полным решением задания вы сможете ознакомиться, нажав на кнопку «Решение»

**Решение 3**

## ЗАДАНИЕ 4

Модифицировать игру «GUESS MY NUMBER» следующим образом:

- пользователь не выбирает уровень, т. к. переход ко второму уровню возможен только после завершения первого без проигрыша (если игрок желает продолжить);
- первый уровень длится три раунда (на каждом раунде загадывается новое число), а второй — два раунда. Новые жизни начисляются игроку перед новым раундом.
- если пользователь проиграл раунд, то компьютеру начисляются очки за раунд в размере начальные жизни

пользователя, умноженные на 5 для первого уровня и на 10 для второго;

- количество очков сохраняется и накапливается в раундах и уровнях;
- промежуточные итоги выводятся после каждого раунда и уровня.

Финальное задание рассчитано на самостоятельное решение. Здесь не будет подсказок и готового алгоритма. Вам необходимо применить все практические навыки, полученные из предыдущих заданий.

**ПОДСКАЗКА К ЗАДАНИЮ 1**

1. Вывод линии представляет собой последовательность (повторение операции) выводов символа, который задал пользователь, определенной длины, поэтому необходимо использовать цикл.
2. Если известно количество повторений (длина линии в символах), какой тип цикла следует использовать?
3. Для формирования вертикальной линии каждый символ линии должен выводиться с новой строки.
4. Для задания скорости отображения линии можно использовать задержку между операциями вывода.
5. Если задержка необходима перед операцией вывода следует ли данную команду помещать в цикл?

**ПОДСКАЗКА К ЗАДАНИЮ 2**

1. Имитация процесса бросания кубика выполняется с помощью генерации случайного числа от 1 до 6 (какая сторона кубика «выпала»).
2. С помощью какой функции в C++ можно сгенерировать случайное число? Необходимо ли подключать дополнительную библиотеку для этого?
3. Какой тип цикла следует использовать для повторения вывода меню (например, в случае ввода несуществующего пункта или после игры)?
4. В зависимости от сгенерированного числа следует отображать соответствующее изображение стороны кубика в консоли.
5. Для определения, кто бросает, кубики первым (пользователь или компьютер) нужно выполнить начальный бросок одного кубика каждым игроком (у кого выпало большее число, тот и ходит первым).
6. Какую встроенную функцию (которая приостанавливает выполнение программы на указанное время) можно использовать для имитации длительности процесса бросания кубика? Необходимо ли подключать дополнительную библиотеку для этого?
7. Каким образом количество раундов (три) связано и циклом, внутри которого реализуются ходы игрока и компьютера?
8. Как определяется победитель и в какой части программы нужно добавлять эту проверку?

**ПОДСКАЗКА К ЗАДАНИЮ 3**

1. Имитация процесса загадывания числа компьютером выполняется с помощью генерации случайного числа в заданном диапазоне.
2. С помощью какой функции в C++ можно сгенерировать случайное число? Необходимо ли подключать дополнительную библиотеку для этого?
3. Какую встроенную функцию (которая приостанавливает выполнение программы на указанное время) можно использовать для имитации длительности процесса загадывания (компьютер «думает» над числом)? Необходимо ли подключать дополнительную библиотеку для этого?
4. Какой тип цикл следует использовать для повторения вывода меню (например, в случае ввода несуществующего пункта или после завершения уровня)?
5. Количество жизней игрока — это определенный % от длины диапазона угадывания. Как вычислить это количество, если диапазон задается двумя числами «от» и «до»?
6. За каждую ошибку игрок штрафуются — минус одна жизнь
7. В какой момент игры пользователю нужно предлагать подсказку?

**РЕШЕНИЕ ЗАДАНИЯ 1****Описание решения**

Вначале необходимо спросить у пользователя символ, на основе которого он желает формировать линию.

Далее переходим к созданию меню, которое будет отображаться пользователю в два этапа: вначале для выбора скорости рисования, затем для выбора типа линии.

Соблюдение такой последовательности этапов отображения и выбора пунктов меню (пользовательских настроек линии) позволит установить задержку между выводами символов, которую далее необходимо использовать уже при рисовании линии (в зависимости от типа, указанного пользователем на втором этапе работы с меню).

Для реализации каждого этапа меню пользователю необходимо вывести номера пунктов меню и их описание, чтобы он ввел нужный ему код пункта меню.

Если пользователь выбрал первый пункт меню (медленно), то устанавливаем большую длительность задержки, например, **1000** (1 секунда).

Если пользователь выбрал второй пункт меню (нормально), то устанавливаем среднюю длительность задержки, например, **500** (0,5 секунды).

Если пользователь выбрал третий пункт меню (быстро), то устанавливаем нулевую длительность задержки, т.е. линия отобразится мгновенно.

Если пользователь ввел непредусмотренный номер пункта меню, то выводим сообщение об ошибке и устанавливаем нулевую длительность задержки (задержка по умолчанию отсутствует).

Далее отображаем пользователю следующую часть меню для выбора типа линии.

Если пользователь выбрал первый пункт меню (вертикальная линия), то организуем цикл, внутри которого выполняются две операции: задержка (установленной длительности) и вывод указанного пользователем символа в консоль с последующим переходом на новую строку.

Если пользователь выбрал второй пункт меню (горизонтальная линия), то организуем цикл, внутри которого выполняются две операции: задержка (установленной длительности) и вывод указанного пользователем символа в консоль без перехода на новую строку.

Если пользователь ввел непредусмотренный номер пункта меню, то выводим сообщение об ошибке и организуем рисование горизонтальной линии (установка по умолчанию).

### Решение

1. Подключаем библиотеку (`#include<windows.h>`), которая позволит в дальнейшем использовать встроенную функцию `Sleep(time)`, которая приостанавливает выполнение программы на время, задаваемое параметром `time`. Выводим строку в консоль с описанием названия программы (по желанию, данная строка кода может быть пропущена), создаем необходимое число переменных: одну символьного типа для хранения символа, из которого будет формироваться линия и три целочисленные для хранения кода типа линии, кода скорости отображения и величины задержки.



```
#include<iostream>
#include<windows.h>
using namespace std;

int main()
{
    cout << "Home task #8.1.1\n\n";
    char userSymbol;
    int lineType, speedID, delay;

    return 0;
}
```

2. Выводим в консоль запрос на символ линии и считываем его в соответствующую переменную.

Далее выводим в консоль описание пунктов первой части меню (выбор скорости отображения линии) и запрашиваем у пользователя номер желаемого пункта, сохраняем его в переменную.

```
#include<iostream>
#include<windows.h>
using namespace std;

int main()
{
    cout << "Home task #8.1.1\n\n";
    char userSymbol;
    int lineType, speedID, delay;

    cout << "Enter a character to make a line:\n";
    cin >> userSymbol;
```

```
cout << "Select the drawing speed:\n";  
cout << "1- slowly \n";  
cout << "2- normally \n";  
cout << "3- quickly \n";  
cin >> speedID;  
  
return 0;  
}
```

3. Реализуем проверку выбранного пользователем номера пункта меню для установки скорости отображения. Будем использовать оператор выбора **switch**, который позволяет организовать проверку одной переменной на соответствие нескольких условий более эффективно и проще, чем набор из нескольких последовательных блоков **if – else if**.

Также после реализации задач выбранного пункта меню нужно остановить процесс проверки по остальным пунктам. Это реализуется путем использования оператора **break** в конце каждого кейса. Кейс **default** обрабатывает ситуацию ввода пользователем недопустимого номера пункта меню.

```
#include<iostream>  
#include<windows.h>  
using namespace std;  
  
int main()  
{  
    cout << "Home task #8.1.1\n\n";  
    char userSymbol;
```

```
int lineType, speedID, delay;

cout << "Enter a character to make a line:\n";
cin >> userSymbol;

cout << "Select the drawing speed:\n";
cout << "1- slowly \n";
cout << "2- normally \n";
cout << "3- quickly \n";
cin >> speedID;

switch (speedID)
{
case 1:
{
    break;
}
case 2:
{
    break;
}
case 3:
{
    break;
}
default:

}

return 0;
}
```

#### 4. Добавим в кейсы установку величины задержки.

Если пользователь выбрал первый пункт меню (медленно), то устанавливаем большую длительность задержки, например, **1000** (1 секунда).

Если пользователь выбрал второй пункт меню (нормально), то устанавливаем среднюю длительность задержки, например, **500** (0,5 секунды).

Если пользователь выбрал третий пункт меню (быстро), то устанавливаем нулевую длительность задержки, т.е. линия отобразится мгновенно.

Если пользователь ввел непредусмотренный номер пункта меню, то выводим сообщение об ошибке и устанавливаем нулевую длительность задержки (задержка по умолчанию отсутствует).

```
#include<iostream>
#include<windows.h>
using namespace std;

int main()
{
    cout << "Home task #8.1.1\n\n";
    char userSymbol;
    int lineType, speedID, delay;

    cout << "Enter a character to make a line:\n";
    cin >> userSymbol;

    cout << "Select the drawing speed:\n";
    cout << "1- slowly \n";
    cout << "2- normally \n";
```

```
cout << "3- quickly \n";
cin >> speedID;

switch (speedID)
{
case 1:
{
    delay = 1000;
    break;
}
case 2:
{
    delay = 500;
    break;
}
case 3:
{
    delay = 0;
    break;
}
default:
    cout << "Wrong speed ID!"<<"\n";
    cout << "The line will be drawn quickly!";
    cout << "\n";
    delay = 0;
}

return 0;
}
```

5. Далее отображаем пользователю следующую часть меню для выбора типа линии.

Проверка пунктов второй части меню организуется аналогично шагу 3 с использованием оператора выбора `switch`.

```
#include <iostream>
#include <windows.h>
using namespace std;

int main()
{
    cout << "Home task #8.1.1\n\n";
    char userSymbol;
    int lineType, speedID, delay;

    cout << "Enter a character to make a line:\n";
    cin >> userSymbol;

    cout << "Select the drawing speed:\n";
    cout << "1- slowly \n";
    cout << "2- normally \n";
    cout << "3- quickly \n";
    cin >> speedID;

    switch (speedID)
    {
        case 1:
        {
            delay = 1000;
            break;
        }
        case 2:
        {
            delay = 500;
            break;
        }
    }
```

```
case 3:
{
    delay = 0;
    break;

}
default:
    cout << "Wrong speed ID!"<<"\n";
    cout << "The line will be drawn quickly!";
    cout << "\n";
    delay = 0;
}
cout << "Select line type:\n";
cout << "1- vertical line \n";
cout << "2- horisontal line \n";
cin >> lineType;
switch (lineType)
{
case 1:
{

    break;

}
case 2:
{

    break;

}

default:
}

return 0;
}
```

## 6. Перейдем к заполнению кейсов.

Если пользователь выбрал первый пункт меню (вертикальная линия), то организуем цикл, внутри которого выполняются две операции: задержка (установленной длительности) и вывод указанного пользователем символа в консоль с последующим переходом на новую строку.

Для примера длина линии будет составлять десять символов.

Если пользователь выбрал второй пункт меню (горизонтальная линия), то организуем цикл, внутри которого выполняются две операции: задержка (установленной длительности) и вывод указанного пользователем символа в консоль без перехода на новую строку.

Если пользователь ввел непредусмотренный номер пункта меню, то выводим сообщение об ошибке и организуем рисование горизонтальной линии (установка по умолчанию).

```
#include<iostream>
#include<windows.h>
using namespace std;

int main()
{
    cout << "Home task #8.1.1\n\n";
    char userSymbol;
    int lineType, speedID, delay;

    cout << "Enter a character to make a line:\n";
    cin >> userSymbol;

    cout << "Select the drawing speed:\n";
    cout << "1- slowly \n";
```



```
cout << "2- normally \n";
cout << "3- quickly \n";
cin >> speedID;

switch (speedID)
{
case 1:
{
    delay = 1000;
    break;
}
case 2:
{
    delay = 500;
    break;
}
case 3:
{
    delay = 0;
    break;
}
default:
    cout << "Wrong speed ID!"<<"\n";
    cout << "The line will be drawn quickly!";
    cout << "\n";
    delay = 0;
}

cout << "Select line type:\n";
cout << "1- vertical line \n";
cout << "2- horisontal line \n";
cin >> lineType;
```

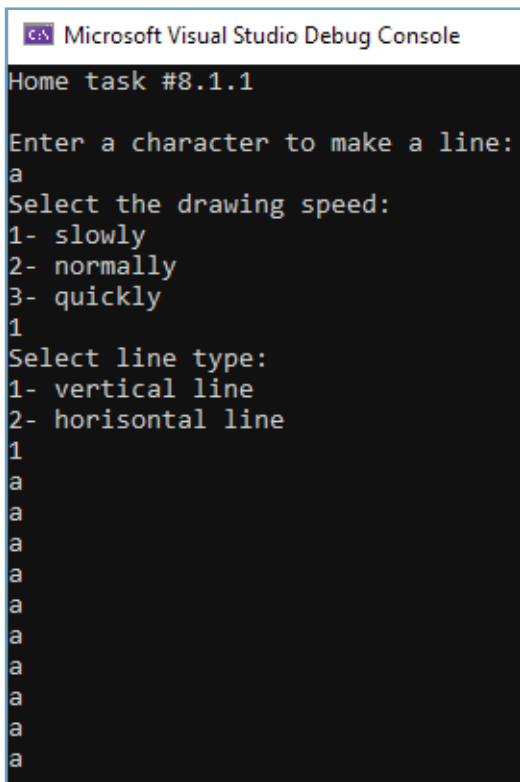
```
switch (lineType)
{
case 1:
{
    for (int i = 0; i < 10; i++)
    {
        Sleep(delay);
        cout << userSymbol << "\n";
    }
    break;
}
case 2:
{
    for (int i = 0; i < 10; i++)
    {
        Sleep(delay);
        cout << userSymbol;
    }
    break;
}
default:
    cout << "Wrong line type!"<<"\n";
    cout << "The horisontal line will be drawn!";
    cout << "\n";

    for (int i = 0; i < 10; i++)
    {
        Sleep(delay);
        cout << userSymbol;
    }
}

return 0;
}
```

Результаты работы программы (в консоли):

- Тест 1 — пользователь выбрал первый пункт первого меню (медленно) и первый пункт второго меню (вертикальная линия)

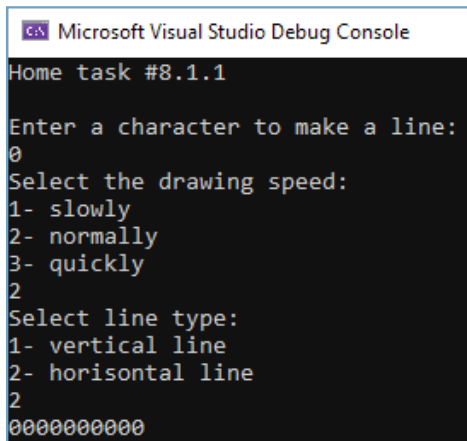


```
Microsoft Visual Studio Debug Console
Home task #8.1.1

Enter a character to make a line:
a
Select the drawing speed:
1- slowly
2- normally
3- quickly
1
Select line type:
1- vertical line
2- horisontal line
1
a
a
a
a
a
a
a
a
a
a
a
```

Рисунок 1

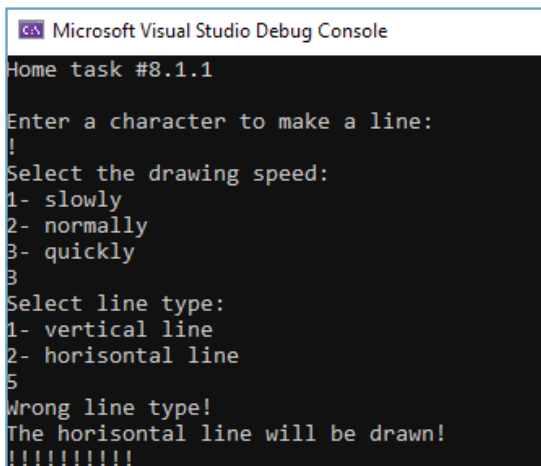
- Тест 2 — пользователь выбрал второй пункт первого меню (нормально) и второй пункт второго меню (горизонтальная линия)



```
Microsoft Visual Studio Debug Console
Home task #8.1.1
Enter a character to make a line:
0
Select the drawing speed:
1- slowly
2- normally
3- quickly
2
Select line type:
1- vertical line
2- horizontal line
2
0000000000
```

Рисунок 2

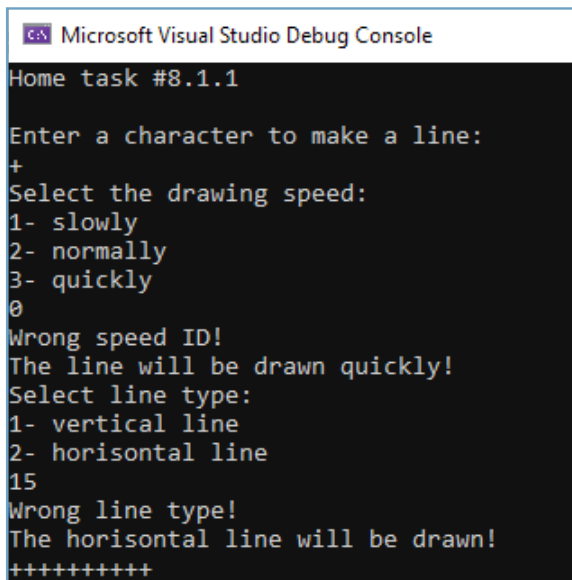
- Тест 3 — пользователь выбрал третий пункт первого меню (быстро) и ввел произвольный символ вместо пункта второго меню



```
Microsoft Visual Studio Debug Console
Home task #8.1.1
Enter a character to make a line:
!
Select the drawing speed:
1- slowly
2- normally
3- quickly
3
Select line type:
1- vertical line
2- horizontal line
5
Wrong line type!
The horizontal line will be drawn!
!!!!!!!!!!!!
```

Рисунок 3

- Тест 4 — пользователь ввел произвольный символ вместо пункта первого меню и ввел произвольный символ вместо пункта второго меню



```
Microsoft Visual Studio Debug Console
Home task #8.1.1

Enter a character to make a line:
+
Select the drawing speed:
1- slowly
2- normally
3- quickly
0
Wrong speed ID!
The line will be drawn quickly!
Select line type:
1- vertical line
2- horisontal line
15
Wrong line type!
The horisontal line will be drawn!
+++++++
```

Рисунок 4

## РЕШЕНИЕ ЗАДАНИЯ 2

### Описание решения

Вначале реализуем цикл для повторения вывода меню игры. Это будет цикл с постусловием, так как вначале необходимо хотя бы раз отобразить меню, а потом проверять, следует ли его повторять. Повторение вывода меню происходит, если пользователь ввел непредусмотренный элемент меню или после игры. Для остановки процесса вывода меню следует предусмотреть пункт «Выход из игры».

Меню будет содержать два пункта: 1 — начать игру; 2 — выход.

В случае, если пользователь выбрал первый пункт меню, определяем, кто будет ходить первым.

Для этого игрок и компьютер по очереди бросают один кубик (генерируем последовательно два числа). У кого выпало большее число, тот первый и ходит.

Также после каждого броска (пользователя или компьютера) нужно выводить в консоль соответствующее изображение выпавшей стороны кубика.

В случае, если у пользователя и компьютера выпало одинаковое число, повторяем пробный бросок, пока не выпадут разные стороны кубика.

Организуем цикл на три раунда.

В каждом раунде по очереди (в зависимости от того, кто ходит первый) игрок и компьютер «бросают» два кубика (генерируем для каждого игрока по два числа в раунде).

Счет (очки) каждого игрока обновляется после каждого раунда, увеличивается на сумму выпавших чисел.

После всех раундов (трех бросков каждого игрока) проверяем, у кого больше очков, тот и является победителем игры. При равенстве очков выводим в консоль сообщение «Ничья».

### Решение

1. Подключаем библиотеку (`#include <windows.h>`), которая позволит в дальнейшем использовать встроенную функцию `Sleep(time)`, которая приостанавливает выполнение программы на указанное время. Таким образом мы сможем имитировать в игре некоторую задержку на процесс бросания кубиков игроком.

Подключаем библиотеку (`#include "time.h"`), которая позволит в дальнейшем использовать встроенную функцию `time()` для получения текущего системного времени, которое необходимо для старта генератора псевдослучайных чисел. Имитация процесса бросания кубика будет выполняться с помощью генерации случайного числа от 1 до 6 (какая сторона кубика «выпала»).

Выводим строку в консоль с описанием названия программы (по желанию, данная строка кода может быть пропущена), создаем шесть целочисленных переменных: одну для хранения кода элемента меню, две для хранения чисел, соответствующих сторонам кубиков и две для хранения очков компьютера и пользователя.

Для того, чтобы при каждом новом запуске программы генерировалось новое число используем функцию `srand()`, а в качестве ее параметра — вызов функции `time(NULL)`. Устанавливаем нулевые начальные значения для счетчиков очков компьютера и пользователя.

```
#include<iostream>
#include <windows.h>
#include<time.h>

using namespace std;

int main()
{
    cout << "Home task #8.1.2 - Dice Game\n";
    int userChoice;
    int Dice1, Dice2;
    int PlayerPoints, BotPoints;
    srand(time(NULL));

    PlayerPoints = 0;
    BotPoints = 0;

    return 0;
}
```

2. Реализуем проверку выбранного пользователем номера пункта меню с помощью оператор выбора `switch`, который позволяет организовать проверку одной переменной на соответствие нескольких условий более эффективно и проще, чем набор из нескольких последовательных блоков `if – else if`.

Также после реализации задач выбранного пункта меню нужно остановить процесс проверки по остальным пунктам. Это реализуется путем использования оператора `break` в конце каждого кейса. Кейс `default` обрабатывает ситуацию ввода пользователем недопустимого номера пункта меню.



```
#include<iostream>
#include <windows.h>
#include<time.h>

using namespace std;

int main()
{
    cout << "Home task #8.1.2 - Dice Game\n";
    int userChoice;
    int Dice1, Dice2;
    int PlayerPoints, BotPoints;
    srand(time(NULL));

    PlayerPoints = 0;
    BotPoints = 0;

    do {
        cout << "Welcome to the Game!\n";
        cout << "Your choice:\n";
        cout << "1 - roll the dice\n";
        cout << "2 - quit\n";
        cin >> userChoice;

        switch (userChoice) {
            case 1:
            {
                break;
            }

            case 2:
            {
                cout << "See you!";
                break;
            }
        }
    }
}
```

```
    }  
    default:  
        cout << "Wrong menu item!";  
    }  
} while (userChoice != 2);  
  
return 0;  
  
}
```

3. В случае, если пользователь выбрал первый пункт меню, определяем, кто будет ходить первым.

Для этого игрок и компьютер по очереди бросают один кубик (генерируем последовательно два числа).

Перед генерацией числа следует использовать встроенную функцию `Sleep(time)`, которая приостанавливает выполнение программы на указанное время. Таким образом мы сможем имитировать в игре некоторую задержку на процесс бросания кубиков игроком

Так как нам не нужны случайные числа между 0 и `RAND_MAX`, которые возвращает функция `rand()` — нам нужны числа между двумя другими значениями, то следует выполнить преобразование полученного от функции `rand()` числа в число из указанного диапазона. Это выполняется с помощью следующей формулы:

$$a + \text{rand}() \% (b - a),$$

где `a` и `b` — границы нужного диапазона.

У кого из игроков выпало большее число, тот первый и ходит.

В случае, если у пользователя и компьютера выпало одинаковое число, повторяем пробный бросок, пока не выпадут разные стороны кубика.

```
#include<iostream>
#include <windows.h>
#include<time.h>

using namespace std;

int main()
{
    cout << "Home task #8.1.2 - Dice Game\n";
    int userChoice;
    int Dice1, Dice2;
    int PlayerPoints, BotPoints;
    srand(time(NULL));

    PlayerPoints = 0;
    BotPoints = 0;
    do {
        cout << "Welcome to the Game!\n";
        cout << "Your choice:\n";
        cout << "1 - roll the dice\n";
        cout << "2 - quit\n";
        cin >> userChoice;

        switch (userChoice) {
            case 1:
            {
                do
                {
                    cout << "Determining the order "
                        "of play\n";
```

```
        cout << "Player is rolling "
              << "the dice...\n";
        Sleep(1000);
        Dice1 = 1 + rand() % 5;
        cout << "Player's result is ";
        cout << Dice1 << "\n";

        cout << "Bot is rolling the dice...\n";
        Sleep(1000);
        Dice2 = 1 + rand() % 5;
        cout << "Bot's result is " << Dice2;
        cout << "\n";
        if (Dice1 == Dice2)
        {
            cout << "Once again.\n";
        }
    } while (Dice1 == Dice2);
    if (Dice1 > Dice2)
    {
        cout << "Player goes first. "
              << "Bot goes second.\n\n";
    }
    else
    {
        cout << "Bot goes first. Player goes "
              << "second.\n\n";
    }

    break;
}

case 2:
{
```

```
        cout << "See you!";  
        break;  
    }  
    default:  
        cout << "Wrong menu item!";  
    }  
} while (userChoice != 2);  
  
return 0;  
}
```

#### 4. Организуем цикл на три раунда.

В каждом раунде по очереди (в зависимости от того, кто ходит первый) игрок и компьютер будут «бросать» два кубика. Делаем небольшую задержку между раундами с помощью функции `Sleep(time)`.

```
#include<iostream>  
#include <windows.h>  
#include<time.h>  
  
using namespace std;  
  
int main()  
{  
    cout << "Home task #8.1.2 - Dice Game\n";  
    int userChoice;  
    int Dice1, Dice2;  
    int PlayerPoints, BotPoints;  
    srand(time(NULL));  
  
    PlayerPoints = 0;
```

```
BotPoints = 0;

do {
    cout << "Welcome to the Game!\n";
    cout << "Your choice:\n";
    cout << "1 - roll the dice\n";
    cout << "2 - quit\n";
    cin >> userChoice;

    switch (userChoice) {
    case 1:
    {
        do
        {
            cout << "Determining the order "
                  "of play\n";
            cout << "Player is rolling "
                  "the dice...\n";
            Sleep(1000);
            Dice1 = 1 + rand() % 5;
            cout << "Player's result is ";
            cout << Dice1 << "\n";
            cout << "Bot is rolling "
                  "the dice...\n";
            Sleep(1000);
            Dice2 = 1 + rand() % 5;
            cout << "Bot's result is " << Dice2;
            cout << "\n";
            if (Dice1 == Dice2)
            {
                cout << "Once again.\n";
            }
        } while (Dice1 == Dice2);
        if (Dice1 > Dice2)
        {
```

```

        cout << "Player goes first. "
              << "Bot goes second.\n\n";
    for (int i = 0; i < 3; i++)
    {
        cout << i + 1 << "-round starts:";
        cout << "\n";
        Sleep(1000);
    }
}
else
{
    cout << "Bot goes first. Player goes";
        << " second.\n\n";
    for (int i = 0; i < 3; i++)
    {
        cout << i + 1 << "-round starts:";
        cout << "\n";
        Sleep(1000);
    }
}

    break;
}
case 2:
{
    cout << "See you!";
    break;
}
default:
    cout << "Wrong menu item!";
}
} while (userChoice != 2);
return 0;
}

```

5. Вначале реализуем ситуация, когда первым ходит пользователь.

В каждом раунде (внутри цикла) при броске пользователя просим подтвердить бросок кубиков вводом числа 1. Не продолжаем игру, пока пользователь не введет данный код продолжения. Используем для реализации этой ситуации цикл `do-while`, так как вначале необходимо хотя бы раз сделать запрос пользователю, а только потом проверить введенный код и повторять запрос при необходимости. После чего организуем задержку (процесс бросания кубиков) и генерируем два случайных числа в диапазоне от единицы до шести включительно. Увеличиваем очки игрока на сумму полученных чисел. Повторяем реализацию «бросков кубиков» таким же образом для компьютера, но без кода подтверждения.

Выводим результаты для каждого раунда.

```
#include<iostream>
#include <windows.h>
#include<time.h>

using namespace std;

int main()
{
    cout << "Home task #8.1.2 - Dice Game\n";
    int userChoice;
    int Dice1, Dice2;
    int PlayerPoints, BotPoints;
    srand(time(NULL));
```



```
PlayerPoints = 0;
BotPoints = 0;

do {
    cout << "Welcome to the Game!\n";
    cout << "Your choice:\n";
    cout << "1 - roll the dice\n";
    cout << "2 - quit\n";
    cin >> userChoice;

    switch (userChoice) {
    case 1:
    {
        do
        {
            cout << "Determining the order "
                  "of play\n";
            cout << "Player is rolling "
                  "the dice...\n";
            Sleep(1000);
            Dice1 = 1 + rand() % 5;
            cout << "Player's result is ";
            cout << Dice1 << "\n";

            cout << "Bot is rolling "
                  "the dice...\n";
            Sleep(1000);
            Dice2 = 1 + rand() % 5;
            cout << "Bot's result is " << Dice2;
            cout << "\n";

            if (Dice1 == Dice2)
            {
                cout << "Once again.\n";
```

```
    }  
    } while (Dice1 == Dice2);  
  
    if (Dice1 > Dice2)  
    {  
  
        cout << "Player goes first. "  
                "Bot goes second.\n\n";  
        for (int i = 0; i < 3; i++)  
        {  
            cout << i + 1 << "-round starts:";  
            cout << "\n";  
            do  
            {  
                cout << "Enter '1' to role "  
                        "the dice.\n";  
                cin >> userChoice;  
            } while (userChoice != 1);  
  
            cout<< "Player is rolling "  
                    "the dice...\n";  
            Sleep(1000);  
            Dice1 = 1 + rand() % 5;  
            Dice2 = 1 + rand() % 5;  
            PlayerPoints = PlayerPoints +  
                            Dice1+ Dice2;  
            cout << "Player " << Dice1<<"-";  
            cout << Dice2 << "\n";  
  
            cout << "Bot is rolling "  
                    "the dice...\n";  
            Sleep(1000);  
            Dice1 = 1 + rand() % 5;
```

```

        Dice2 = 1 + rand() % 5;
        BotPoints = BotPoints + Dice1 +
            Dice2;
        cout << "Bot " << Dice1 << "-";
        cout << Dice2 << "\n";
        cout << i+1<< "-round results:";
        cout << "\n";
        cout << "PlayerPoints - ";
        cout << PlayerPoints << "\n";
        cout << "BotPoints - ";
        cout << BotPoints << "\n\n";
        Sleep(1000);
    }
}
else
{
    cout << "Bot goes first. "
        "Player goes second.\n\n";
    for (int i = 0; i < 3; i++)
    {
        cout << i + 1;
        cout << "-round starts:" << "\n";
        Sleep(1000);
    }
}

break;
}
case 2:
{
    cout << "See you!";
    break;
}
default:

```

```
        cout << "Wrong menu item!";  
    }  
} while (userChoice != 2);  
  
return 0;  
}
```

6. Аналогично шагу 5 реализуется ситуация, когда первым ходит компьютер.

```
#include<iostream>  
#include <windows.h>  
#include<time.h>  
  
using namespace std;  
  
int main()  
{  
    cout << "Home task #8.1.2 - Dice Game\n";  
    int userChoice;  
    int Dice1, Dice2;  
    int PlayerPoints, BotPoints;  
    srand(time(NULL));  
  
    PlayerPoints = 0;  
    BotPoints = 0;  
  
    do {  
        cout << "Welcome to the Game!\n";  
        cout << "Your choice:\n";  
        cout << "1 - roll the dice\n";  
        cout << "2 - quit\n";
```

```
cin >> userChoice;

switch (userChoice) {
case 1:
{
    do
    {
        cout << "Determining the order "
              "of play\n";
        cout << "Player is rolling "
              "the dice...\n";
        Sleep(1000);
        Dice1 = 1 + rand() % 5;
        cout << "Player's result is ";
        cout << Dice1 << "\n";

        cout << "Bot is rolling "
              "the dice...\n";
        Sleep(1000);
        Dice2 = 1 + rand() % 5;
        cout << "Bot's result is ";
        cout << Dice2 << "\n";
        if (Dice1 == Dice2)
        {
            cout << "Once again.\n";
        }
    } while (Dice1 == Dice2);
    if (Dice1 > Dice2)
    {
        cout << "Player goes first. "
              "Bot goes second.\n\n";
        for (int i = 0; i < 3; i++)
        {
```

```

cout << i + 1 << "-round starts:";
cout << "\n";
do
{
    cout << "Enter '1' to role "
           "the dice.\n";
    cin >> userChoice;
} while (userChoice != 1);

cout<< "Player is rolling "
       "the dice...\n";
Sleep(1000);
Dice1 = 1 + rand() % 5;
Dice2 = 1 + rand() % 5;
PlayerPoints = PlayerPoints +
               Dice1+ Dice2;
cout << "Player " << Dice1<<"-";
cout << Dice2 << "\n";

cout << "Bot is rolling "
       "the dice...\n";
Sleep(1000);
Dice1 = 1 + rand() % 5;
Dice2 = 1 + rand() % 5;
BotPoints = BotPoints + Dice1 +
            Dice2;
cout << "Bot " << Dice1 << "-";
cout << Dice2 << "\n";
cout << i+1<< "-round results:";
cout << "\n";
cout << "PlayerPoints - ";
cout << PlayerPoints << "\n";
cout << "BotPoints - ";
cout << BotPoints << "\n\n";

```

```
        Sleep(1000);  
    }  
}  
else  
{  
    cout << "Bot goes first. "  
        "Player goes second.\n\n";  
    for (int i = 0; i < 3; i++)  
    {  
        cout << i + 1 << "-round starts:";  
        cout << "\n";  
        cout << "Bot is rolling "  
            "the dice...\n";  
        Sleep(1000);  
        Dice1 = 1 + rand() % 5;  
        Dice2 = 1 + rand() % 5;  
        BotPoints = BotPoints + Dice1 +  
            Dice2;  
        cout << "Bot " << Dice1 << "-";  
        cout << Dice2 << "\n";  
  
        do  
        {  
            cout << "Enter '1' to role "  
                "the dice.\n";  
            cin >> userChoice;  
        } while (userChoice != 1);  
  
        cout << "Player is rolling "  
            "the dice...\n";  
        Sleep(1000);  
        Dice1 = 1 + rand() % 5;  
        Dice2 = 1 + rand() % 5;
```

```
        PlayerPoints = PlayerPoints +  
                        Dice1 + Dice2;  
        cout << "Player " << Dice1;  
        cout << "-" << Dice2 << "\n";  
  
        cout << i + 1 << "-round results:";  
        cout << "\n";  
        cout << "BotPoints  - ";  
        cout << BotPoints << "\n\n";  
        cout << "PlayerPoints - ";  
        cout << PlayerPoints << "\n";  
        Sleep(1000);  
    }  
    }  
    break;  
}  
  
case 2:  
{  
    cout << "See you!";  
    break;  
}  
  
default:  
    cout << "Wrong menu item!";  
}  
} while (userChoice != 2);  
  
return 0;  
}
```

7. После окончания трех раундов (завершения цикла) выводим результаты игра, определив победителя или ситуацию «Ничья».



```
#include<iostream>
#include <windows.h>
#include<time.h>

using namespace std;

int main()
{
    cout << "Home task #8.1.2 - Dice Game\n";
    int userChoice;
    int Dice1, Dice2;
    int PlayerPoints, BotPoints;
    srand(time(NULL));

    PlayerPoints = 0;
    BotPoints = 0;

    do {
        cout << "Welcome to the Game!\n";
        cout << "Your choice:\n";
        cout << "1 - roll the dice\n";
        cout << "2 - quit\n";
        cin >> userChoice;

        switch (userChoice) {
            case 1:
            {
                do
                {
                    cout << "Determining the order "
                        "of play\n";
                    cout << "Player is rolling "
                        "the dice...\n";
                    Sleep(1000);
```

```

Dice1 = 1 + rand() % 5;
cout << "Player's result is "
      << Dice1 << "\n";

cout << "Bot is rolling "
      << "the dice...\n";
Sleep(1000);
Dice2 = 1 + rand() % 5;
cout << "Bot's result is " << Dice2;
cout << "\n";

if (Dice1 == Dice2)
{
    cout << "Once again.\n";
}
} while (Dice1 == Dice2);

if (Dice1 > Dice2)
{
    cout << "Player goes first. "
          << "Bot goes second.\n\n";
    for (int i = 0; i < 3; i++)
    {
        cout << i + 1 << "-round starts:";
        cout << "\n";
        do
        {
            cout << "Enter '1' to role "
                  << "the dice.\n";
            cin >> userChoice;
        } while (userChoice != 1);

        cout<< "Player is rolling "
              << "the dice...\n";
    }
}

```

```

        Sleep(1000);
        Dice1 = 1 + rand() % 5;
        Dice2 = 1 + rand() % 5;
        PlayerPoints = PlayerPoints +
                        Dice1+ Dice2;
        cout << "Player " << Dice1<<"-";
        cout << Dice2 << "\n";

        cout << "Bot is rolling "
                "the dice...\n";
        Sleep(1000);
        Dice1 = 1 + rand() % 5;
        Dice2 = 1 + rand() % 5;
        BotPoints = BotPoints + Dice1 +
                        Dice2;
        cout << "Bot " << Dice1 << "-";
        cout << Dice2 << "\n";

        cout << i+1<< "-round results:";
        cout << "\n";
        cout << "PlayerPoints - ";
        cout << PlayerPoints << "\n";
        cout << "BotPoints - ";
        cout << BotPoints << "\n\n";

        Sleep(1000);
    }
}
else
{
    cout << "Bot goes first. "
            "Player goes second.\n\n";
    for (int i = 0; i < 3; i++)
    {

```

```
cout << i + 1 << "-round starts:";
cout << "\n";
cout << "Bot is rolling "
      "the dice...\n";
Sleep(1000);
Dice1 = 1 + rand() % 5;
Dice2 = 1 + rand() % 5;
BotPoints = BotPoints + Dice1 +
            Dice2;
cout << "Bot " << Dice1 << "-";
cout << Dice2 << "\n";

do
{
    cout << "Enter '1' to role "
          "the dice.\n";
    cin >> userChoice;
} while (userChoice != 1);

cout << "Player is rolling "
      "the dice...\n";
Sleep(1000);
Dice1 = 1 + rand() % 5;
Dice2 = 1 + rand() % 5;
PlayerPoints = PlayerPoints +
              Dice1 + Dice2;
cout << "Player " << Dice1;
cout << "-" << Dice2 << "\n";

cout << i + 1 << "-round results:";
cout << "\n";
cout << "BotPoints  - ";
cout << BotPoints << "\n\n";
```

```
        cout << "PlayerPoints - ";
        cout << PlayerPoints << "\n";
        Sleep(1000);
    }
}

cout << "Game results:" << "\n";
cout << "PlayerPoints - ";
cout << PlayerPoints << "\n";
cout << "BotPoints - " << BotPoints;
cout << "\n\n";
if (PlayerPoints > BotPoints)
{
    cout << "The player is the winner!";
    cout << "\n";
}
else if (PlayerPoints < BotPoints)
{
    cout << "The bot is the winner!";
    cout << "\n";
}
else
{
    cout << "The draw!" << "\n";
}
break;
}

case 2:
{
    cout << "See you!";
    break;
}

default:
    cout << "Wrong menu item!";
```

```
    }  
    } while (userChoice != 2);  
  
    return 0;  
}
```

Добавляем отображение выпавших сторон кубиков в консоль после каждого броска. Реализуем проверку выпавшего числа (какое изображение выводить) с помощью оператор выбора `switch`. Также после реализации отображения кубика нужно остановить процесс проверки по остальным пунктам. Это реализуется путем использования оператора `break` в конце каждого кейса. Кейс `default` нам не понадобится, так как проверяемое число было сгенерировано в диапазоне от единицы до шести.

```
#include<iostream>  
#include <windows.h>  
#include<time.h>  
  
using namespace std;  
  
int main()  
{  
    cout << "Home task #8.1.2 - Dice Game\n";  
    int userChoice;  
    int Dice1, Dice2;  
    int PlayerPoints, BotPoints;  
    srand(time(NULL));  
  
    PlayerPoints = 0;
```

```
BotPoints = 0;

do {
    cout << "Welcome to the Game!\n";
    cout << "Your choice:\n";
    cout << "1 - roll the dice\n";
    cout << "2 - quit\n";
    cin >> userChoice;

    switch (userChoice) {
    case 1:
    {
        do
        {
            cout << "Determining the order "
                  "of play\n";
            cout << "Player is rolling "
                  "the dice...\n";
            Sleep(1000);
            Dice1 = 1 + rand() % 5;
            cout << "Player's result is ";
            cout << Dice1 << "\n";
            switch (Dice1) {
            case 1:
            {
                cout << "=====" << "\n";
                cout << "|           |" << "\n";
                cout << "|           |" << "\n";
                cout << "|      *      |" << "\n";
                cout << "|           |" << "\n";
                cout << "|           |" << "\n";
                cout << "=====" << "\n";
                break;
            }
        }
    }
}
```

```

case 2:
{
    cout << "=====" << "\n";
    cout << "|         |" << "\n";
    cout << "|      *      |" << "\n";
    cout << "|         |" << "\n";
    cout << "|      *      |" << "\n";
    cout << "|         |" << "\n";
    cout << "=====" << "\n";
    break;
}
case 3:
{
    cout << "=====" << "\n";
    cout << "|         |" << "\n";
    cout << "|      *      |" << "\n";
    cout << "|      *      |" << "\n";
    cout << "|         *    |" << "\n";
    cout << "|         |" << "\n";
    cout << "=====" << "\n";
    break;
}
case 4:
{
    cout << "=====" << "\n";
    cout << "|         |" << "\n";
    cout << "|      *      *      |" << "\n";
    cout << "|         |" << "\n";
    cout << "|      *      *      |" << "\n";
    cout << "|         |" << "\n";
    cout << "=====" << "\n";
    break;
}

```



```

case 5:
{
    cout << "=====" << "\n";
    cout << "|           |" << "\n";
    cout << "|  *   *  |" << "\n";
    cout << "|    *    |" << "\n";
    cout << "|  *   *  |" << "\n";
    cout << "|           |" << "\n";
    cout << "=====" << "\n";
    break;
}
case 6:
{
    cout << "=====" << "\n";
    cout << "|           |" << "\n";
    cout << "|  *   *  |" << "\n";
    cout << "|  *   *  |" << "\n";
    cout << "|  *   *  |" << "\n";
    cout << "|           |" << "\n";
    cout << "=====" << "\n";
    break;
}

cout << "Bot is rolling "
      "the dice...\n";
Sleep(1000);
Dice2 = 1 + rand() % 5;
cout << "Bot's result is " << Dice2;
cout << "\n";
switch (Dice2) {
case 1:
{

```

```

        cout << "=====" << "\n";
        cout << " | " << "\n";
        cout << " | " << "\n";
        cout << " | * | " << "\n";
        cout << " | " << "\n";
        cout << " | " << "\n";
        cout << "=====" << "\n";
        break;
    }
    case 2:
    {
        cout << "=====" << "\n";
        cout << " | " << "\n";
        cout << " | * | " << "\n";
        cout << " | " << "\n";
        cout << " | * | " << "\n";
        cout << " | " << "\n";
        cout << "=====" << "\n";
        break;
    }
    case 3:
    {
        cout << "=====" << "\n";
        cout << " | " << "\n";
        cout << " | * | " << "\n";
        cout << " | * | " << "\n";
        cout << " | * | " << "\n";
        cout << " | " << "\n";
        cout << "=====" << "\n";
        break;
    }
    case 4:
    {

```

```

        cout << "=====" << "\n";
        cout << "|           |" << "\n";
        cout << "|    *    *    |" << "\n";
        cout << "|           |" << "\n";
        cout << "|    *    *    |" << "\n";
        cout << "|           |" << "\n";
        cout << "=====" << "\n";
        break;
    }
    case 5:
    {
        cout << "=====" << "\n";
        cout << "|           |" << "\n";
        cout << "|    *    *    |" << "\n";
        cout << "|        *      |" << "\n";
        cout << "|    *    *    |" << "\n";
        cout << "|           |" << "\n";
        cout << "=====" << "\n";
        break;
    }
    case 6:
    {
        cout << "=====" << "\n";
        cout << "|           |" << "\n";
        cout << "|    *    *    |" << "\n";
        cout << "|    *    *    |" << "\n";
        cout << "|    *    *    |" << "\n";
        cout << "|           |" << "\n";
        cout << "=====" << "\n";
        break;
    }
}
if (Dice1 == Dice2)
{

```

```
        cout << "Once again.\n";
    }
} while (Dice1 == Dice2);

if (Dice1 > Dice2)
{
    cout << "Player goes first. "
           "Bot goes second.\n\n";
    for (int i = 0; i < 3; i++)
    {
        cout << i + 1 << "-round starts:";
        cout << "\n";
        do
        {
            cout << "Enter '1' to role "
                   "the dice.\n";
            cin >> userChoice;
        } while (userChoice != 1);
        cout << "Player is rolling "
               "the dice...\n";
        Sleep(1000);
        Dice1 = 1 + rand() % 5;
        Dice2 = 1 + rand() % 5;
        PlayerPoints = PlayerPoints +
                       Dice1+ Dice2;
        cout << "Player's result is ";
        cout << Dice1<<"-"<< Dice2 << "\n";
        switch (Dice1) {
        case 1:
        {
            cout << "=====" << "\n";
            cout << " |           | " << "\n";
            cout << " |           | " << "\n";
```

```

        cout << " |      *      |" << "\n";
        cout << " |              |" << "\n";
        cout << " |              |" << "\n";
        cout << "===== " << "\n";
        break;
    }
    case 2:
    {
        cout << "===== " << "\n";
        cout << " |              |" << "\n";
        cout << " |      *      |" << "\n";
        cout << " |              |" << "\n";
        cout << " |      *      |" << "\n";
        cout << " |              |" << "\n";
        cout << "===== " << "\n";
        break;
    }
    case 3:
    {
        cout << "===== " << "\n";
        cout << " |              |" << "\n";
        cout << " |      *      |" << "\n";
        cout << " |      *      |" << "\n";
        cout << " |              * |" << "\n";
        cout << " |              |" << "\n";
        cout << "===== " << "\n";
        break;
    }
    case 4:
    {
        cout << "===== " << "\n";
        cout << " |              |" << "\n";
        cout << " |      *      * |" << "\n";
    }

```

```

        cout << " |           | " << "\n";
        cout << " |      *      * | " << "\n";
        cout << " |           | " << "\n";

        cout << "===== " << "\n";
        break;
    }
    case 5:
    {
        cout << "===== " << "\n";
        cout << " |           | " << "\n";
        cout << " |      *      * | " << "\n";
        cout << " |          *      | " << "\n";
        cout << " |      *      * | " << "\n";
        cout << " |           | " << "\n";
        cout << "===== " << "\n";
        break;
    }
    case 6:
    {
        cout << "===== " << "\n";
        cout << " |           | " << "\n";
        cout << " |      *      * | " << "\n";
        cout << " |      *      * | " << "\n";
        cout << " |      *      * | " << "\n";
        cout << " |           | " << "\n";
        cout << "===== " << "\n";
        break;
    }
}

switch (Dice2) {
case 1:

```

```

    {
        cout << "=====" << "\n";
        cout << " |      | " << "\n";
        cout << " |      | " << "\n";
        cout << " |      *      | " << "\n";
        cout << " |      |      | " << "\n";
        cout << " |      |      | " << "\n";
        cout << "=====" << "\n";
        break;
    }
case 2:
{
    cout << "=====" << "\n";
    cout << " |      | " << "\n";
    cout << " |      *      | " << "\n";
    cout << " |      |      | " << "\n";
    cout << " |      *      | " << "\n";
    cout << " |      |      | " << "\n";
    cout << "=====" << "\n";
    break;
}
case 3:
{
    cout << "=====" << "\n";
    cout << " |      | " << "\n";
    cout << " |      *      | " << "\n";
    cout << " |      *      | " << "\n";
    cout << " |      *      | " << "\n";
    cout << " |      |      | " << "\n";
    cout << "=====" << "\n";
    break;
}

```

```

case 4:
{
    cout << "=====" << "\n";
    cout << "|      |" << "\n";
    cout << "|  *  *  |" << "\n";
    cout << "|      |" << "\n";
    cout << "|  *  *  |" << "\n";
    cout << "|      |" << "\n";
    cout << "=====" << "\n";
    break;
}
case 5:
{
    cout << "=====" << "\n";
    cout << "|      |" << "\n";
    cout << "|  *  *  |" << "\n";
    cout << "|    *    |" << "\n";
    cout << "|  *  *  |" << "\n";
    cout << "|      |" << "\n";
    cout << "=====" << "\n";
    break;
}
case 6:
{
    cout << "=====" << "\n";
    cout << "|      |" << "\n";
    cout << "|  *  *  |" << "\n";
    cout << "|  *  *  |" << "\n";
    cout << "|  *  *  |" << "\n";
    cout << "|      |" << "\n";
    cout << "=====" << "\n";
    break;
}

```



```

    }

    cout << "Bot is rolling "
           "the dice...\n";

    Sleep(1000);
    Dice1 = 1 + rand() % 5;
    Dice2 = 1 + rand() % 5;
    BotPoints = BotPoints + Dice1 +
                Dice2;
    cout << "Bot's result is ";
    cout << Dice1 << "-" << Dice2;
    cout << "\n";

    switch (Dice1) {
    case 1:
    {
        cout << "=====" << "\n";
        cout << " |           | " << "\n";
        cout << " |           | " << "\n";
        cout << " |      *      | " << "\n";
        cout << " |           | " << "\n";
        cout << " |           | " << "\n";
        cout << "=====" << "\n";
        break;
    }
    case 2:
    {
        cout << "=====" << "\n";
        cout << " |           | " << "\n";
        cout << " |      *      | " << "\n";
        cout << " |           | " << "\n";
        cout << " |      *      | " << "\n";
        cout << " |           | " << "\n";
        cout << "=====" << "\n";
    }
    }
    }

```

```

        break;
    }
    case 3:
    {
        cout << "===== " << "\n";
        cout << " |           | " << "\n";
        cout << " |      *      | " << "\n";
        cout << " |         *      | " << "\n";
        cout << " |            *      | " << "\n";
        cout << " |           | " << "\n";
        cout << "===== " << "\n";
        break;
    }
    case 4:
    {
        cout << "===== " << "\n";
        cout << " |           | " << "\n";
        cout << " |      *      *      | " << "\n";
        cout << " |           | " << "\n";
        cout << " |      *      *      | " << "\n";
        cout << " |           | " << "\n";
        cout << "===== " << "\n";
        break;
    }
    case 5:
    {
        cout << "===== " << "\n";
        cout << " |           | " << "\n";
        cout << " |      *      *      | " << "\n";
        cout << " |         *      | " << "\n";
        cout << " |      *      *      | " << "\n";
        cout << " |           | " << "\n";
        cout << "===== " << "\n";
        break;
    }

```

```

    }
    case 6:
    {
        cout << "===== " << "\n";
        cout << " |           | " << "\n";
        cout << " |   *   *   | " << "\n";
        cout << " |   *   *   | " << "\n";
        cout << " |   *   *   | " << "\n";
        cout << " |           | " << "\n";
        cout << "===== " << "\n";
        break;
    }
}

switch (Dice2) {
case 1:
{
    cout << "===== " << "\n";
    cout << " |           | " << "\n";
    cout << " |           | " << "\n";
    cout << " |       *       | " << "\n";
    cout << " |           | " << "\n";
    cout << " |           | " << "\n";
    cout << "===== " << "\n";
    break;
}
case 2:
{
    cout << "===== " << "\n";
    cout << " |           | " << "\n";
    cout << " |       *       | " << "\n";
    cout << " |           | " << "\n";
    cout << " |       *       | " << "\n";

```

```

        cout << " |          | " << "\n";
        cout << "===== " << "\n";
        break;
    }
    case 3:
    {
        cout << "===== " << "\n";
        cout << " |          | " << "\n";
        cout << " |      *      | " << "\n";
        cout << " |          *      | " << "\n";
        cout << " |              *      | " << "\n";
        cout << " |          | " << "\n";
        cout << "===== " << "\n";
        break;
    }
    case 4:
    {
        cout << "===== " << "\n";
        cout << " |          | " << "\n";
        cout << " |      *      *      | " << "\n";
        cout << " |          | " << "\n";
        cout << " |      *      *      | " << "\n";
        cout << " |          | " << "\n";
        cout << "===== " << "\n";
        break;
    }
    case 5:
    {
        cout << "===== " << "\n";
        cout << " |          | " << "\n";
        cout << " |      *      *      | " << "\n";
        cout << " |          *      | " << "\n";
        cout << " |      *      *      | " << "\n";
    }

```

```

        cout << " |           | " << "\n";
        cout << "===== " << "\n";
        break;
    }
    case 6:
    {
        cout << "===== " << "\n";
        cout << " |           | " << "\n";
        cout << " | *       * | " << "\n";
        cout << " | *       * | " << "\n";
        cout << " | *       * | " << "\n";
        cout << " |           | " << "\n";
        cout << "===== " << "\n";
        break;
    }

    }

    cout << i+1 << "-round results:";
    cout << "\n";
    cout << "PlayerPoints - ";
    cout << PlayerPoints << "\n";
    cout << "BotPoints - ";
    cout << BotPoints << "\n\n";
    Sleep(1000);
}
}
else
{
    cout << "Bot goes first. "
           "Player goes second.\n\n";
    for (int i = 0; i < 3; i++)
    {

```

```

cout << i + 1;
cout << "-round starts:" << "\n";
cout << "Bot is rolling the dice...\n";
Sleep(1000);
Dice1 = 1 + rand() % 5;
Dice2 = 1 + rand() % 5;
BotPoints = BotPoints + Dice1 +
            Dice2;
cout << "Bot's result is ";
cout << Dice1 << "-" << Dice2;
cout << "\n";
switch (Dice1) {
case 1:
{
    cout << "=====" << "\n";
    cout << "|          |" << "\n";
    cout << "|          |" << "\n";
    cout << "|      *      |" << "\n";
    cout << "|          |" << "\n";
    cout << "|          |" << "\n";
    cout << "=====" << "\n";
    break;
}
case 2:
{
    cout << "=====" << "\n";
    cout << "|          |" << "\n";
    cout << "|      *      |" << "\n";
    cout << "|          |" << "\n";
    cout << "|      *      |" << "\n";
    cout << "|          |" << "\n";
    cout << "=====" << "\n";
    break;
}
}

```

```

case 3:
{
    cout << "===== " << "\n";
    cout << " |           | " << "\n";
    cout << " |      *      | " << "\n";
    cout << " |         *      | " << "\n";
    cout << " |          *      | " << "\n";
    cout << " |           | " << "\n";
    cout << "===== " << "\n";
    break;
}
case 4:
{
    cout << "===== " << "\n";
    cout << " |           | " << "\n";
    cout << " |      *      *      | " << "\n";
    cout << " |           | " << "\n";
    cout << " |      *      *      | " << "\n";
    cout << " |           | " << "\n";
    cout << "===== " << "\n";
    break;
}
case 5:
{
    cout << "===== " << "\n";
    cout << " |           | " << "\n";
    cout << " |      *      *      | " << "\n";
    cout << " |         *      | " << "\n";
    cout << " |      *      *      | " << "\n";
    cout << " |           | " << "\n";
    cout << "===== " << "\n";
    break;
}
case 6:
{

```

```

        cout << "=====" << "\n";
        cout << "|          |" << "\n";
        cout << "|      *      |" << "\n";
        cout << "|      *      |" << "\n";
        cout << "|      *      |" << "\n";
        cout << "|          |" << "\n";
        cout << "=====" << "\n";
        break;
    }
}
switch (Dice2) {
case 1:
{
    cout << "=====" << "\n";
    cout << "|          |" << "\n";
    cout << "|          |" << "\n";
    cout << "|      *      |" << "\n";
    cout << "|          |" << "\n";
    cout << "|          |" << "\n";
    cout << "=====" << "\n";
    break;
}
case 2:
{
    cout << "=====" << "\n";
    cout << "|          |" << "\n";
    cout << "|      *      |" << "\n";
    cout << "|          |" << "\n";
    cout << "|      *      |" << "\n";
    cout << "|          |" << "\n";
    cout << "=====" << "\n";
    break;
}
case 3:
{

```



```

        cout << "=====" << "\n";
        cout << "|      |" << "\n";
        cout << "| *      |" << "\n";
        cout << "|      *    |" << "\n";
        cout << "|          * |" << "\n";
        cout << "|          |" << "\n";
        cout << "=====" << "\n";
        break;
    }
    case 4:
    {
        cout << "=====" << "\n";
        cout << "|      |" << "\n";
        cout << "| *    * |" << "\n";
        cout << "|      |" << "\n";
        cout << "| *    * |" << "\n";
        cout << "|      |" << "\n";
        cout << "=====" << "\n";
        break;
    }
    case 5:
    {
        cout << "=====" << "\n";
        cout << "|      |" << "\n";
        cout << "| *    * |" << "\n";
        cout << "|      * |" << "\n";
        cout << "| *    * |" << "\n";
        cout << "|      |" << "\n";
        cout << "=====" << "\n";
        break;
    }
    case 6:
    {
        cout << "=====" << "\n";

```

```

        cout << "|           |" << "\n";
        cout << "|    *    *    |" << "\n";
        cout << "|    *    *    |" << "\n";
        cout << "|    *    *    |" << "\n";
        cout << "|           |" << "\n";
        cout << "===== " << "\n";
        break;
    }
}
do
{
    cout << "Enter '1' to role "
           "the dice.\n";
    cin >> userChoice;
} while (userChoice != 1);
cout << "Player is rolling "
       "the dice...\n";
Sleep(1000);
Dice1 = 1 + rand() % 5;
Dice2 = 1 + rand() % 5;
PlayerPoints = PlayerPoints +
               Dice1 + Dice2;
cout << "Player's result is ";
cout << Dice1 << "-" << Dice2;
cout << "\n";
switch (Dice1) {
case 1:
{
    cout << "===== " << "\n";
    cout << "|           |" << "\n";
    cout << "|           |" << "\n";
    cout << "|    *    |" << "\n";
    cout << "|           |" << "\n";
    cout << "|           |" << "\n";

```

```

        cout << "======" << "\n";
        break;
    }
    case 2:
    {
        cout << "======" << "\n";
        cout << "|          |" << "\n";
        cout << "|      *      |" << "\n";
        cout << "|          |" << "\n";
        cout << "|      *      |" << "\n";
        cout << "|          |" << "\n";
        cout << "======" << "\n";
        break;
    }
    case 3:
    {
        cout << "======" << "\n";
        cout << "|          |" << "\n";
        cout << "|      *      |" << "\n";
        cout << "|          *      |" << "\n";
        cout << "|          *      |" << "\n";
        cout << "|          |" << "\n";
        cout << "======" << "\n";
        break;
    }
    case 4:
    {
        cout << "======" << "\n";
        cout << "|          |" << "\n";
        cout << "|      *      *      |" << "\n";
        cout << "|          |" << "\n";
        cout << "|      *      *      |" << "\n";
        cout << "|          |" << "\n";
        cout << "======" << "\n";
    }

```

```

        break;
    }
    case 5:
    {
        cout << "=====" << "\n";
        cout << "|         |" << "\n";
        cout << "|  *   *  |" << "\n";
        cout << "|    *    |" << "\n";
        cout << "|  *   *  |" << "\n";
        cout << "|         |" << "\n";
        cout << "=====" << "\n";
        break;
    }
    case 6:
    {
        cout << "=====" << "\n";
        cout << "|         |" << "\n";
        cout << "|  *   *  |" << "\n";
        cout << "|  *   *  |" << "\n";
        cout << "|  *   *  |" << "\n";
        cout << "|         |" << "\n";
        cout << "=====" << "\n";
        break;
    }
}
switch (Dice2) {
case 1:
{
    cout << "=====" << "\n";
    cout << "|         |" << "\n";
    cout << "|         |" << "\n";
    cout << "|    *    |" << "\n";
    cout << "|         |" << "\n";
    cout << "|         |" << "\n";
}
}

```

```

        cout << "======" << "\n";
        break;
    }
    case 2:
    {
        cout << "======" << "\n";
        cout << "|          |" << "\n";
        cout << "|      *      |" << "\n";
        cout << "|          |" << "\n";
        cout << "|      *      |" << "\n";
        cout << "|          |" << "\n";
        cout << "======" << "\n";
        break;
    }
    case 3:
    {
        cout << "======" << "\n";
        cout << "|          |" << "\n";
        cout << "|      *      |" << "\n";
        cout << "|      *      |" << "\n";
        cout << "|          *   |" << "\n";
        cout << "|          |" << "\n";
        cout << "======" << "\n";
        break;
    }
    case 4:
    {
        cout << "======" << "\n";
        cout << "|          |" << "\n";
        cout << "|      *      *   |" << "\n";
        cout << "|          |" << "\n";
        cout << "|      *      *   |" << "\n";
        cout << "|          |" << "\n";
        cout << "======" << "\n";
    }

```

```

        break;
    }
    case 5:
    {
        cout << "======" << "\n";
        cout << "|      |" << "\n";
        cout << "|  *  *  |" << "\n";
        cout << "|    *    |" << "\n";
        cout << "|  *  *  |" << "\n";
        cout << "|      |" << "\n";
        cout << "======" << "\n";
        break;
    }
    case 6:
    {
        cout << "======" << "\n";
        cout << "|      |" << "\n";
        cout << "|  *  *  |" << "\n";
        cout << "|  *  *  |" << "\n";
        cout << "|  *  *  |" << "\n";
        cout << "|      |" << "\n";
        cout << "======" << "\n";
        break;
    }
}

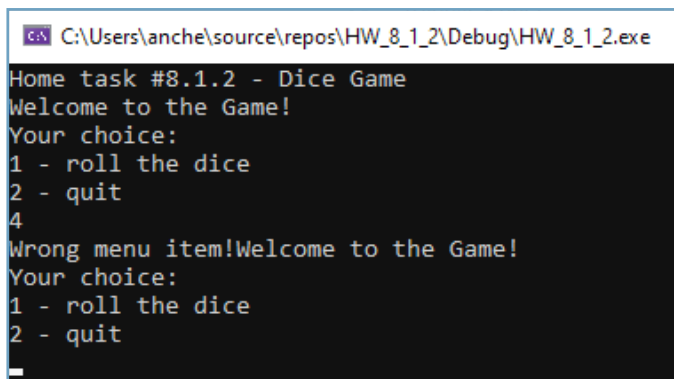
cout << i + 1;
cout << "-round results:";
cout << "\n";
cout << "BotPoints  - ";
cout << BotPoints;
cout << "\n\n";
cout << "PlayerPoints  - ";
cout << PlayerPoints << "\n";

```

```
        Sleep(1000);
    }
}
cout << "Game results:" << "\n";
cout << "PlayerPoints - ";
cout << PlayerPoints << "\n";
cout << "BotPoints - " << BotPoints;
cout << "\n\n";
if (PlayerPoints > BotPoints)
{
    cout << "The player is the winner!";
    cout << "\n";
}
else if (PlayerPoints < BotPoints)
{
    cout << "The bot is the winner!";
    cout << "\n";
}
else
{
    cout << "The draw!" << "\n";
}
break;
}
case 2:
{
    cout << "See you!";
    break;
}
default:
    cout << "Wrong menu item!";
}
} while (userChoice != 2);
return 0;
}
```

Результаты работы программы в консоли.

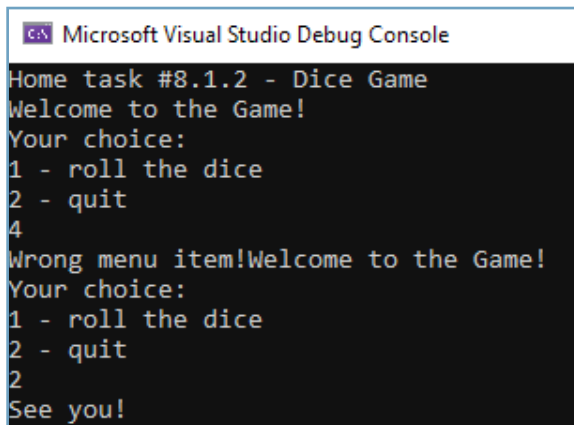
- Тест 1 — пользователь ввел код несуществующего пункта меню.



```
C:\Users\anche\source\repos\HW_8_1_2\Debug\HW_8_1_2.exe
Home task #8.1.2 - Dice Game
Welcome to the Game!
Your choice:
1 - roll the dice
2 - quit
4
Wrong menu item!Welcome to the Game!
Your choice:
1 - roll the dice
2 - quit
_
```

Рисунок 5

- Тест 2 — пользователь ввел код выхода из игры.



```
C:\> Microsoft Visual Studio Debug Console
Home task #8.1.2 - Dice Game
Welcome to the Game!
Your choice:
1 - roll the dice
2 - quit
2
Wrong menu item!Welcome to the Game!
Your choice:
1 - roll the dice
2 - quit
2
See you!
```

Рисунок 6



- Тест 3 — при пробном броске выпали одинаковые результаты.

```
C:\Users\anche\source\repos\HW_8_1_2\Debug\HW_8_1_2.exe
Home task #8.1.2 - Dice Game
Welcome to the Game!
Your choice:
1 - roll the dice
2 - quit
1
Determining the order of play
Player is rolling the dice...
Player's result is 2
=====
  *
  *
=====
Bot is rolling the dice...
Bot's result is 2
=====
  *
  *
=====
Once again.
```

Рисунок 7

- Тест 4 — продолжаем игру, по пробному броску первым выпадает ходить боту.

```
Determining the order of play
Player is rolling the dice...
Player's result is 3
=====
|
| *
|  *
|   *
|
|-----|
Bot is rolling the dice...
Bot's result is 4
=====
|
| *  *
| *  *
|
|-----|
Bot goes first. Player goes second.
```

Рисунок 8


- Тест 5 — продолжаем игру, бот бросил кубики, а пользователь ввел неверный код подтверждения своего броска.

```
1-round starts:
Bot is rolling the dice...
Bot's result is 5-2
=====
| * * |
| * |
| * * |
|=====|
|=====|
| * |
| * |
|=====|
Enter '1' to role the dice.
2
Enter '1' to role the dice.
_
```


Рисунок 9

- Тест 6 — продолжаем игру, пользователь ввел верный код подтверждения своего броска.

```
Enter '1' to role the dice.  
1  
Player is rolling the dice...  
Player's result is 1-5  
  
=====
```



```
=====
```



```
=====
```

1-round results:  
BotPoints - 7  
PlayerPoints - 6

Рисунок 10

- Тест 7 — продолжаем игру, второй раунд

```
2-round starts:  
Bot is rolling the dice...  
Bot's result is 1-2  
  
=====
```

| |

\*

| |

```
=====
```

| |

\*

\*

| |

```
=====
```

Enter '1' to role the dice.  
1  
Player is rolling the dice...  
Player's result is 1-2  
  
=====

| |

\*

| |

```
=====
```

| |

\*

\*

| |

```
=====
```

Рисунок 11

```
2-round results:
BotPoints - 10

PlayerPoints - 9
```

Рисунок 12

- Тест 8 — продолжаем игру, третий раунд

```
3-round starts:
Bot is rolling the dice...
Bot's result is 4-5

=====
| *  * |
| *  * |
|=====|
| *  * |
|  *  |
| *  * |
|=====|
Enter '1' to role the dice.
1
Player is rolling the dice...
Player's result is 3-2

=====
| *    |
|  *   |
|    * |
|=====|
| *    |
|  *   |
|=====|
```

Рисунок 13

```
3-round results:  
BotPoints - 19  
  
PlayerPoints - 14  
Game results:  
PlayerPoints - 14  
BotPoints - 19  
  
The bot is the winner!  
Welcome to the Game!  
Your choice:  
1 - roll the dice  
2 - quit  
2  
See you!
```

Рисунок 14

## РЕШЕНИЕ ЗАДАНИЯ 3

## Описание решения

Вначале реализуем цикл для повторения вывода меню игры. Это будет цикл с постусловием, так как вначале необходимо хотя бы раз отобразить меню, а потом проверять, следует ли его повторять. Повторение вывода меню происходит, если пользователь ввел непредусмотренный элемент меню или после игры. Для остановки процесса вывода меню следует предусмотреть пункт «Выход из игры».

Меню будет содержать три пункта: 1 — первый уровень сложности; 2 — второй уровень сложности; 3 — выход.

В случае, если пользователь выбрал первый или второй пункт меню, определяем начальное количество жизней игрока по формуле:

$$\text{lives} = (b - a + 1) / n, \quad (1)$$

где  $a, b$  — границы диапазона угадывания;

$n$  — это число 2 для первого уровня (50%) и 4 для второго (25%).

Количество жизней — это целое число, поэтому будем использовать оператор получения целой части от деления.

Например, для первого уровня диапазон угадывания от 10 до 100:

$$\text{lives} = (100 - 10 + 1) / 4 = 24.$$

Далее компьютер «загадывает число» — выполняем генерацию числа в нужном диапазоне соответственно уровню игры.

Игра длится, пока игрок жив (т. е. количество жизней больше нуля) или пока игрок не угадает число.

Организуем цикл с постусловием, так как вначале необходимо хотя бы раз предложить игроку угадать, а потом



проверять, продолжается ли игра (не выполнилось ни одно из условий остановки игры).

В цикле предлагаем пользователю угадать число и сравниваем его с загаданным.

Если введенное пользователем число равно загаданному, то игра останавливается (выполняется одно из условий остановки цикла — игрок угадал).

Иначе «штрафуем» игрока, уменьшая количество жизней на одну.

Далее предлагаем пользователю получить подсказку. Если пользователь согласен, то уменьшаем количество жизней игрока на одну (стоимость подсказки) и выводим в консоль сообщение «Загаданное число больше твоего» или «Загаданное число меньше твоего» в зависимости от ситуации.

После окончания цикла (не зависимо от того, по какой причине остановилась игра) выводим результат игры.

Для определения результата игры вначале проверяем, «жив» ли игрок (количество жизней больше нуля). Если нет, то результат игры — «Проигрыш», иначе вычисляем очки: количество оставшихся жизней, умноженные на 5 для первого уровня и на 10 для второго.

### Решение

1. Подключаем библиотеку (`#include<windows.h>`), которая позволит в дальнейшем использовать встроенную функцию `Sleep(time)`, которая приостанавливает выполнение программы на указанное время. Таким образом мы сможем имитировать в игре некоторую задержку на процесс «загадывания» компьютером числа.

Подключаем библиотеку (`#include "time.h"`), которая позволит в дальнейшем использовать встроенную функцию `time()` для получения текущего системного времени, которое необходимо для старта генератора псевдослучайных чисел. Имитация процесса загадывания числа будет выполняться с помощью генерации случайного числа в диапазоне соответственно уровню игры.

Выводим строку в консоль с описанием названия программы (по желанию, данная строка кода может быть пропущена), создаем девять целочисленных переменных для хранения:

- номера уровня игры;
- границ диапазона загадываемого числа;
- загаданного числа;
- числа, вводимого пользователем (в процессе угадывания);
- кода элемента меню;
- количества жизней пользователя;
- очки пользователя;
- признак было ли угадано число (состояние процесса угадывания).

Для того, чтобы при каждом новом запуске программы генерировалось новое число используем функцию `srand()`, а в качестве ее параметра — вызов функции `time(NULL)`. Устанавливаем нулевые начальные значения для очков пользователя и состояния процесса угадывания (0 — число не угадано, 1 — число угадано).

```
#include<iostream>
#include <windows.h>
#include<time.h>
```

```
using namespace std;

int main()
{
    cout << "Home task #8.1.3 - Magic Number Game\n";
    int level, a, b, magicNum, userNum;
    int userChoice, userLives, userPoints, guessed;
    userPoints = 0;
    guessed = 0;
    srand(time(NULL));

    return 0;
}
```

2. Реализуем цикл для повторения вывода меню игры. Это будет цикл с постусловием, так как вначале необходимо хотя бы раз отобразить меню, а потом проверять, следует ли его повторять. Повторение вывода меню происходит, если пользователь ввел непредусмотренный элемент меню или после игры. Для остановки процесса вывода меню следует предусмотреть пункт «Выход из игры».

Меню будет содержать три пункта: 1 — первый уровень сложности; 2 — второй уровень сложности; 3 — выход.

Реализуем проверку выбранного пользователем номера пункта меню с помощью оператор выбора `switch`, который позволяет организовать проверку одной переменной на соответствие нескольких условий более эффективно и проще, чем набор из нескольких последовательных блоков `if – else if`.

Также после реализации задач выбранного пункта меню нужно остановить процесс проверки по остальным пунктам. Это реализуется путем использования оператора `break` в конце каждого кейса. Кейс `default` обрабатывает ситуацию ввода пользователем недопустимого номера пункта меню.

```
#include<iostream>
#include <windows.h>
#include<time.h>

using namespace std;

int main()
{
    cout << "Home task #8.1.3 - Magic Number Game\n";
    int level, a, b, magicNum, userNum;
    int userChoice, userLives, userPoints, guessed;
    userPoints = 0;
    guessed = 0;
    srand(time(NULL));

    do {
        cout << "Welcome to the Game!\n";
        cout << "Your choice:\n";
        cout << "1 - 1st level: magic number [1..10]\n";
        cout << "2 - 2nd level: magic number [10..100]\n";
        cout << "3 - quit\n";
        cin >> userChoice;
        switch (userChoice) {
            case 1:
            {
                cout << "Welcome to the 1st2nd level: "
                    << "magic number [1..10]!";
```

```
        break;
    }

    case 2:
    {
        cout << "Welcome to the 2nd level: "
              << "magic number [10..100]!";
        break;
    }

    case 3:
    {
        cout << "See you!";
        break;
    }

    default:
        cout << "Wrong menu item!";
    }
} while (userChoice != 3);

return 0;
}
```

3. В случае, если пользователь выбрал первый или второй пункт меню, определяем начальное количество жизней игрока по формуле (1)

```
#include<iostream>
#include <windows.h>
#include<time.h>

using namespace std;
```

```
int main()
{
    cout << "Home task #8.1.3 - Magic Number Game\n";
    int level, a, b, magicNum, userNum;
    int userChoice, userLives, userPoints, guessed;
    userPoints = 0;
    guessed = 0;
    srand(time(NULL));

    do {
        cout << "Welcome to the Game!\n";
        cout << "Your choice:\n";
        cout << "1 - 1st level: magic number "
                "[1..10]\n";
        cout << "2 - 2nd level: magic number "
                "[10..100]\n";
        cout << "3 - quit\n";
        cin >> userChoice;

        switch (userChoice) {
            case 1:
            {
                cout << "Welcome to the 1st level: "
                        "magic number [1..10]! \n";
                a = 1;
                b = 10;
                userLives = (b - a + 1) / 2;
                break;
            }

            case 2:
            {
                cout << "Welcome to the 2nd level: "
                        "magic number [10..100]! \n";
```

```
        a = 10;
        b = 100;
        userLives = (b - a + 1) / 4;

        break;
    }

    case 3:
    {
        cout << "See you!";
        break;
    }
    default:
        cout << "Wrong menu item!";
    }
    while (userChoice != 3);

    return 0;
}
```

4. Далее компьютер «загадывает число» — выполняем генерацию числа в нужном диапазоне соответственно уровню игры. Так как функция `rand()` генерирует следующее случайное число в от 0 до `RANDMAX` (константа, значением которой является 32767), а нам нужно число из диапазона  $[a;b]$ , где  $a$  и  $b$  — это границы диапазона в зависимости от уровня игры, то следует выполнить преобразование полученного от функции `rand()` числа в число из указанного диапазона. Это выполняется с помощью следующей формулы:

$$a + \text{rand}() \% (b - a),$$

где  $a$  и  $b$  — границы нужного диапазона.

Инициализация переменных **a** и **b** происходит соответствующими значениями в зависимости от уровня игры (кейс 1 или кейс 2).

Игра длится, пока игрок жив (т.е. количество жизней больше нуля) или пока игрок не угадает число.

Организуем цикл с постусловием, так как вначале необходимо хотя бы раз предложить игроку угадать, а потом проверять, продолжается ли игра (не выполнилось ни одно из условий остановки игры).

В цикле предлагаем пользователю угадать число (ввести его с консоли).

```
#include<iostream>
#include <windows.h>
#include<time.h>

using namespace std;

int main()
{
    cout << "Home task #8.1.3 - Magic Number Game\n";
    int level, a, b, magicNum, userNum;
    int userChoice, userLives, userPoints, guessed;
    userPoints = 0;
    guessed = 0;
    srand(time(NULL));

    do {
        cout << "Welcome to the Game!\n";
        cout << "Your choice:\n";
        cout << "1 - 1st level: magic number "
             << "[1..10]\n";
```



```
cout << "2 - 2nd level: magic number "  
        "[10..100]\n";  
cout << "3 - quit\n";  
cin >> userChoice;  
  
switch (userChoice) {  
case 1:  
{  
    cout << "Welcome to the 1st level: "  
            "magic number [1..10]!\n";  
    a = 1;  
    b = 10;  
    userLives = (b - a + 1) / 2;  
    cout << "Bot thinks a number...\n";  
    Sleep(1000);  
    magicNum = a + rand() % (b-a);  
    cout << "Magic number is ready! Let's "  
            "try to guess it!\n";  
    do {  
        cout << "Try to guess, your "  
                "naumber?\n";  
        cin >> userNum;  
    } while ((userLives > 0) && (guessed == 0));  
    break;  
}  
  
case 2:  
{  
    cout << "Welcome to the 2nd level: "  
            "magic number [10..100]!\n";  
    a = 10;  
    b = 100;  
    userLives = (b - a + 1) / 4;
```

```
        cout << "Bot thinks a number...\n";
        Sleep(1000);
        magicNum = a + rand() % (b - a);
        cout << "Magic number is ready! "
                "Let's try to guess it!\n";

        do {
            cout << "Try to guess, "
                    "your naumber?\n";
            cin >> userNum;
        } while ((userLives > 0) && (guessed == 0));

        break;
    }

    case 3:
    {
        cout << "See you!";
        break;
    }
    default:
        cout << "Wrong menu item!";
    }
} while (userChoice != 3);

return 0;
}
```

5. Если введенное пользователем число равно загаданному, то игра останавливается (выполняется одно из условий остановки цикла — игрок угадал). Устанавливаем значение переменной **guessed** (состояния угадывания) в единицу, чтобы остановить цикл игры.

```
#include<iostream>
#include <windows.h>
#include<time.h>

using namespace std;

int main()
{
    cout << "Home task #8.1.3 - Magic Number Game\n";
    int level, a, b, magicNum, userNum;
    int userChoice, userLives, userPoints, guessed;
    userPoints = 0;
    guessed = 0;
    srand(time(NULL));

    do {
        cout << "Welcome to the Game!\n";
        cout << "Your choice:\n";
        cout << "1 - 1st level: magic number [1..10]\n";
        cout << "2 - 2nd level: magic number [10..100]\n";
        cout << "3 - quit\n";
        cin >> userChoice;

        switch (userChoice) {
            case 1:
            {
                cout << "Welcome to the 1st level: "
                    << "magic number [1..10]!\n";
                a = 1;
                b = 10;
                userLives = (b - a + 1) / 2;
                cout << "Bot thinks a number...\n";
                Sleep(1000);
```

```
magicNum = a + rand() % (b-a);
cout << "Magic number is ready! "
      "Let's try to guess it!\n";

do {
    cout << "Try to guess, your naumber?\n";
    cin >> userNum;

    if (userNum == magicNum)
    {
        cout << "You guessed magic number!\n";
        guessed = 1;
    }

} while ((userLives > 0) && (guessed == 0));

break;
}

case 2:
{
    cout << "Welcome to the 2nd level: "
          "magic number [10..100]!\n";
    a = 10;
    b = 100;
    userLives = (b - a + 1) / 4;
    cout << "Bot thinks a number...\n";
    Sleep(1000);
    magicNum = a + rand() % (b - a);
    cout << "Magic number is ready! "
          "Let's try to guess it!\n";
    do {
        cout << "Try to guess, your naumber?\n";
        cin >> userNum;
```

```
        if (userNum == magicNum)
        {
            cout << "You guessed magic number!\n";
            guessed = 1;
        }
    } while ((userLives > 0) && (guessed == 0));

    break;
}

case 3:
{
    cout << "See you!";
    break;
}

default:
    cout << "Wrong menu item!";
}

} while (userChoice != 3);

return 0;
}
```

6. Иначе «штрафуем» игрока, уменьшая количество жизней на одну.

Далее предлагаем пользователю получить подсказку.

Если пользователь согласен, то уменьшаем количество жизней игрока на одну (стоимость подсказки) и выводим в консоль сообщение «Загаданное число больше твоего» или «Загаданное число меньше твоего» в зависимости от ситуации.

Организация процесса предложения подсказки также реализуется в виде небольшого меню с помощью цикла `do-while`.

После окончания цикла (не зависимо от того, по какой причине остановилась игра) выводим результат игры.

Для определения результата игры вначале проверяем значение переменной `userPoints`. Если значение равно нулю (т.е. количество жизней равно нулю), то результат игры — «Проигрыш», иначе вычисляем очки: количество оставшихся жизней, умноженные на 5 для первого уровня и на 10 для второго.

```
#include<iostream>
#include <windows.h>
#include<time.h>

using namespace std;

int main()
{
    cout << "Home task #8.1.3 - Magic Number Game\n";
    int level, a, b, magicNum, userNum;
    int userChoice, userLives, userPoints, guessed;
    userPoints = 0;
    guessed = 0;
    srand(time(NULL));

    do {
        cout << "Welcome to the Game!\n";
        cout << "Your choice:\n";
        cout << "1 - 1st level: magic number [1..10]\n";
        cout << "2 - 2nd level: magic number [10..100]\n";
```

```
cout << "3 - quit\n";
cin >> userChoice;
switch (userChoice) {
case 1:
{
    cout << "Welcome to the 1st level: "
           "magic number [1..10]!\n";
    a = 1;
    b = 10;
    userLives = (b - a + 1) / 2;
    cout << "Bot thinks a number...\n";
    Sleep(1000);
    magicNum = a + rand() % (b-a);
    cout << "Magic number is ready! "
           "Let's try to guess it!\n";
    do {
        cout << "Just now you have ";
        cout << userLives << "lives.\n";
        cout << "Try to guess, your number?\n";
        cin >> userNum;
        if (userNum == magicNum)
        {
            cout << "You guessed magic number!\n";
            guessed = 1;
        }
        else {
            cout << "Oh! I lost 1 life!\n";
            userLives--;
            cout << "Just now you have ";
            cout << userLives << "\n";
            do {
                cout << "Do you wish a little "
                       "hint? It costs "
                       "1 life!\n";
```

```
        cout << "Your choice:\n";
        cout << "1 - yes\n";
        cout << "0 - no\n";
        cin >> userChoice;
        if (userChoice == 1)
        {
            if (userNum > magicNum)
            {
                cout << "Let's give "
                    << "less...\n";
                userLives--;
            }
            else
            {
                cout << "Let's give "
                    << "more...\n";
                userLives--;
            }
        }
        } while ((userChoice != 1) &&
                (userChoice != 0));
    }
} while ((userLives > 0) && (guessed == 0));
userPoints = userLives * 5;
if (userPoints == 0)
{
    cout << "You lose!\n";
}
else
{
    cout << "Your score:" << userPoints;
    cout << "\n";
}
break;
```



```
}  
case 2:  
{  
    cout << "Welcome to the 2nd level: "  
           "magic number [10..100]!\n";  
    a = 10;  
    b = 100;  
    userLives = (b - a + 1) / 4;  
    cout << "Bot thinks a number...\n";  
    Sleep(1000);  
    magicNum = a + rand() % (b - a);  
    cout << "Magic number is ready! "  
           "Let's try to guess it!\n";  
    do {  
        cout << "Just now you have ";  
        cout << userLives << "lives.\n";  
        cout << "Try to guess, your number?\n";  
        cin >> userNum;  
        if (userNum == magicNum)  
        {  
            cout << "You guessed magic number!\n";  
            guessed = 1;  
        }  
        else {  
            cout << "Oh! I lost 1 life!\n";  
            userLives--;  
            cout << "Just now you have ";  
            cout << userLives << "\n";  
            do {  
                cout << "Do you wish a little "  
                       "hint? It costs "  
                       "1 life!\n";  
                cout << "Your choice:\n";  
                cout << "1 - yes\n";
```

```
        cout << "0 - no\n";
        cin >> userChoice;
        if (userChoice == 1)
        {
            if (userNum > magicNum)
            {
                cout << "Let's give "
                    << "less...\n";
                userLives--;
            }
            else
            {
                cout << "Let's give "
                    << "more...\n";
                userLives--;
            }
        }
    } while ((userChoice != 1) &&
            (userChoice != 0));
}
} while ((userLives > 0) && (guessed == 0));
userPoints = userLives * 10;
if (userPoints == 0)
{
    cout << "You lose!\n";
}
else
{
    cout << "Your score:" << userPoints;
    cout << "\n";
}
break;
}
```

```
        case 3:
        {
            cout << "See you!";
            break;
        }
        default:
            cout << "Wrong menu item!";
        }
    } while (userChoice != 3);

    return 0;
}
```

Результаты работы программы в консоли.

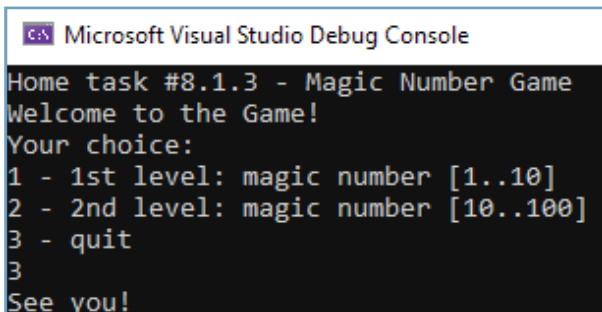
- Тест 1 — пользователь ввел код несуществующего пункта меню.

C:\Users\anche\source\repos\HW\_8\_1\_3\Debug\HW\_8\_1\_3.exe

```
Home task #8.1.3 - Magic Number Game
Welcome to the Game!
Your choice:
1 - 1st level: magic number [1..10]
2 - 2nd level: magic number [10..100]
3 - quit
5
Wrong menu item!Welcome to the Game!
Your choice:
1 - 1st level: magic number [1..10]
2 - 2nd level: magic number [10..100]
3 - quit
```

Рисунок 15

- Тест 2 — пользователь ввел код выхода из игры.

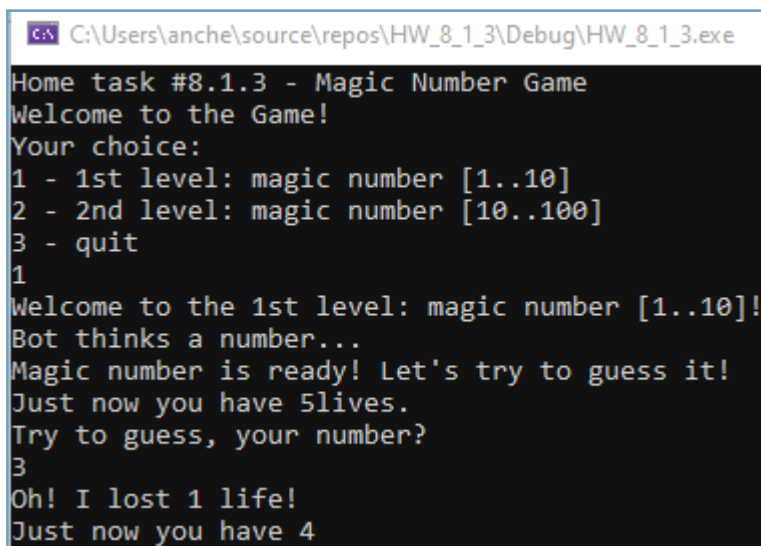


```
C:\> Microsoft Visual Studio Debug Console

Home task #8.1.3 - Magic Number Game
Welcome to the Game!
Your choice:
1 - 1st level: magic number [1..10]
2 - 2nd level: magic number [10..100]
3 - quit
3
See you!
```

Рисунок 16

- Тест 3 — пользователь выбрал первый уровень игры, попробовал угадать число и не угадал.



```
C:\Users\anche\source\repos\HW_8_1_3\Debug\HW_8_1_3.exe

Home task #8.1.3 - Magic Number Game
Welcome to the Game!
Your choice:
1 - 1st level: magic number [1..10]
2 - 2nd level: magic number [10..100]
3 - quit
1
Welcome to the 1st level: magic number [1..10]!
Bot thinks a number...
Magic number is ready! Let's try to guess it!
Just now you have 5lives.
Try to guess, your number?
3
Oh! I lost 1 life!
Just now you have 4
```

Рисунок 17

- Тест 4 — продолжаем игру, пользователю выводится предложение о подсказке, пользователь ввел недопустимый код решения.

```
Oh! I lost 1 life!  
Just now you have 4  
Do you wish a little hint? It costs 1 life!  
Your choice:  
1 - yes  
0 - no  
7  
Do you wish a little hint? It costs 1 life!  
Your choice:  
1 - yes  
0 - no
```

Рисунок 18

- Тест 5 — продолжаем игру, пользователю опять выводится предложение о подсказке, и он соглашается на подсказку.

```
Do you wish a little hint? It costs 1 life!  
Your choice:  
1 - yes  
0 - no  
1  
Let's give less...  
Just now you have 3lives.  
Try to guess, your number?  
-
```

Рисунок 19

- Тест 6 — продолжаем игру, пользователь угадал число.

```
Try to guess, your number?  
2  
You guessed magic number!  
Your score:15  
Welcome to the Game!  
Your choice:  
1 - 1st level: magic number [1..10]  
2 - 2nd level: magic number [10..100]  
3 - quit
```

Рисунок 20

- Тест 7 — продолжаем игру, пользователь выбирает второй уровень, пользователь угадывает число.

```
Welcome to the 2nd level: magic number [10..100]!  
Bot thinks a number...  
Magic number is ready! Let's try to guess it!  
Just now you have 22lives.  
Try to guess, your number?  
34  
You guessed magic number!  
Your score:220  
Welcome to the Game!  
Your choice:  
1 - 1st level: magic number [1..10]  
2 - 2nd level: magic number [10..100]  
3 - quit
```

Рисунок 21