



# **Khulna University of Engineering & Technology**

---

## **Hangman Game with C++ and Object- Oriented Programming**

---

**Submitted by**

**Saleheen Uddin Sakin**

**Dept: CSE**

**Roll: 2107103**

**Group: B2**

**Year: 1<sup>st</sup>**

**Semester: 2<sup>nd</sup>**

**Course Code: CSE1206**

**Course Title: Object-Oriented Programming Laboratory**

## Objectives

- To develop a program using C++ programming Language
- To implement all the Object-Oriented Programming concepts in a program
- To learn how to develop and debug programs in a project

## Introduction

Hangman is a simple word guessing game. Players try to figure out an unknown word by guessing letters. If too many letters which do not appear in the word are entered, the player is hanged (loses).

The "Hangman Game with C++ and Object-Oriented Programming" project is a console-based(text) game. This project illustrates the use of Object-Oriented Programming (OOP) principles to develop an interactive game. The purpose of the game program is to display the usage of C++ and OOP ideas. Some of the used concepts are:

**Inheritance:** Inheritance is utilized in the code to create a derived class, `playerClass`, inheriting properties and methods from a base class, `parentPlayer`. This concept helps code organization and code to reuse.

**Polymorphism:** The code applies polymorphism through virtual functions. The `setPlayerName`, `setPlayerScore`, and `getPlayerName` functions are declared as virtual functions in the base class and overridden in the derived class, demonstrating polymorphic behavior. The `parentPlayer` class represents an abstract class, serving as a blueprint for player classes. It defines pure virtual functions, compelling derived classes to implement specific methods.

**Encapsulation:** The program encapsulates player-related data and methods within the `parentPlayer` and `playerClass` classes. This concept ensures data privacy and maintains a clean interface for interacting with player objects.

**File Handling:** File handling is demonstrated in the code to save score of every player and retrieve high scores from a file (`scores.txt`).

**Friend Class:** The concept of friend classes to enable mutual access to private members of classes. The `friend` keyword is used to establish a relation between the `checkInput` function and the `parentPlayer` and `playerClass` classes. This allows the `checkInput` function to access and manipulate private member data of those classes.

**STL (Standard Template Library):** The concepts of Standard Template Library (STL) is used to manage data efficiently. One of the STL containers `std::set`, is used to store and manage used letters in the game timeline.

## Code

```
#include <bits/stdc++.h>
#include <cstdlib>
#include <ctime>
#include <chrono>
#include <thread>
#include <fstream>

using namespace std;

class parentPlayer
{
protected:
    string playerName;
    int playerScore;
public:
    virtual void setPlayerName(string name)=0;
    virtual void setPlayerScore(int a)=0;
    virtual string getPlayerName()=0;
    friend void checkInput(char inputByPlayer,int *wrongTurn,string &secretWord,string
    &userWord,string &wordClue,int ifMatch,parentPlayer *ob1,int *remainingLife,set<char>&
    mySet,long long int *timeGone);
};

class playerClass:public parentPlayer
{
protected:
public:
    playerClass()
    {
        playerName="Default User";
        playerScore=100;
    }
    void setPlayerName(string name)
    {
        playerName=name;
        playerScore=100;
    }
    void setPlayerScore(int a)
    {
```

```

        playerScore=a;
    }
    string getPlayerName()
    {
        return playerName;
    }
    friend void checkInput(char inputByPlayer,int *wrongTurn,string &secretWord,string
&userWord,string &wordClue,int ifMatch,parentPlayer *ob1,int *remainingLife,set<char>&
mySet,long long int *timeGone);
};

class secretWordClass
{
    string secretWordList[26]=
    {
        "APPLE", "BANANA", "CAT", "DOG", "ELEPHANT", "FLOWER", "GUITAR",
"HONEY", "ISLAND", "JACKET",
        "KITE", "LEMON", "MOON", "NEST", "ORANGE", "PEAR", "QUEEN", "ROSE",
"STAR", "TREE",
        "UMBRELLA", "VASE", "WATERMELON", "XYLOPHONE", "YACHT", "ZEBRA"
    };
    string wordClueList[26]=
    {
        "A common fruit.", "A popular tropical fruit.", "A small domesticated carnivorous
mammal.",
        "A domesticated mammal known for loyalty.", "A large mammal with a trunk and tusks.",
        "A colorful and fragrant plant part.", "A musical instrument with strings.",
        "A sweet, sticky substance made by bees.", "A piece of land surrounded by water.",
        "A type of clothing worn on the upper body.", "A flying toy that dances in the wind.",
        "A sour yellow citrus fruit.", "A natural satellite that orbits the Earth.",
        "A bird's home made from twigs and leaves.", "A round, citrus fruit with a tough outer
layer.",
        "A sweet and juicy fruit.", "A female ruler or monarch.",
        "A type of flower known for its beauty and fragrance.", "A luminous celestial object in the
night sky.",
        "A tall woody plant with branches and leaves.", "A protective device used in the rain.",
        "A container often used for holding flowers.", "A large juicy fruit with green skin and red
flesh.",
        "A musical instrument with wooden bars.", "A luxurious boat often used for sailing.",
        "A striped African animal known for its distinctive pattern.",
    };
    string generatedWord;
public:
    string generateSecretWord(int *remainingLifebar)
    {

        int max;
        max = 25;
        srand(time(0));
        *remainingLifebar=rand()%max;
    }

```

```

        generatedWord=secretWordList[*remainingLifebar];
        return generatedWord;
    }
    string generateWordClue(int remainingLifebar)
    {
        return wordClueList[remainingLifebar];
    }
    string generateUserWord()
    {
        string userWord(generatedWord.size(), '_');
        return userWord;
    }
};

```

```

void checkInput(char inputByPlayer,int *wrongTurn,string &secretWord,string
&userWord,string &wordClue,int ifMatch,parentPlayer *ob1,int *remainingLife,set<char>&
mySet,long long int *timeGone)

```

```

{
    ifMatch=0;
    if(inputByPlayer >= 'a' && inputByPlayer<='z')
    {
        inputByPlayer-=32;
    }
}

```

```

    string givenPunishment[7]= {"=====\n||  |\n||  |\n||  |\n||  |\n||\n
n=====", "=====\n||  |\n||  |\n||  0\n||  |\n||\n
n=====", "=====\n||  |\n||  |\n||  0\n||  /\n||  |\n||\n
n=====", "=====\n||  |\n||  |\n||  0\n||  /\n||  |\n||\n
n=====", "=====\n||  |\n||  |\n||  0\n||  /\n||  /\n||\n
n=====", "=====\n||  |\n||  |\n||  0\n||  /\n||  /\n||\n
n====="};

```

```

for(int i=0; i<secretWord.size(); i++)
{
    if(inputByPlayer==secretWord[i])
    {
        userWord[i]=inputByPlayer;
        ifMatch=1;
    }
}
if(ifMatch==0)
{
    int ltrChk=0;
    for (const char& usedLetter : mySet)
    {
        if(usedLetter==inputByPlayer)
        {
            ltrChk=1;

```

```

        break;
    }
}
if(ltrChk==0)
{
    ob1->playerScore-=ceil(100/6);
    if(ob1->playerScore==4)
    {
        ob1->playerScore=0;
    }
    (*wrongTurn)++;
}
}

cout<<endl<<givenPunishment[*wrongTurn]<<endl<<"Clue:
"<<wordClue<<endl<<"Timeline: "<<userWord<<endl<<"Used Letters: ";

mySet.insert(inputByPlayer);

for (const char& usedLetter : mySet)
{
    std::cout << usedLetter << " ";
}

cout<<"\nCurrent Score: "<<ob1->playerScore<<endl<<"Time remaining: "<<120-
*timeGone<<endl<<endl;

if (userWord == secretWord)
{
    *remainingLife=0;
    cout<<"Congo, "<<ob1->playerName<<"! Your Score is: "<<ob1->playerScore<<endl;

    ofstream fout;
    fout.open("scores.txt", std::ios_base::app);
    fout << ob1->playerName <<endl<<ob1->playerScore<<endl;
    fout.close();
}
else if(*wrongTurn==6)
{
    *remainingLife=0;
    cout<<"You are dead, "<<ob1->playerName<<"! The word was: "<<secretWord<<endl;
}
else if(*timeGone>=120)
{
    cout<<"Time is over, "<<ob1->playerName<<"! The word was: "<<secretWord<<endl;
}
}
}

```

```

void firstShow(string userWord,string wordClue)
{
    cout<<"\n\n===== \n||  \n||  \n||  \n||  \n|| \n===== \n"<<"Clue:
"<<wordClue<<endl<<"Timeline: "<<userWord<<endl<<"Time stars now... You have only
120seconds."<<endl<<endl;
}

void showHighScore()
{
    ifstream fin;
    fin.open("scores.txt");
    string hsName,tempname;
    int highScore=0,tempscore;
    while(fin.eof()==0)
    {
        fin>>tempname>>tempscore;
        if(tempscore>=highScore)
        {
            highScore=tempscore;
            hsName=tempname;
        }
    }
    if(highScore>0)
    {
        cout<<"\nLatest Highest Scorer: "<<hsName<<endl<<"Highest Score:
"<<highScore<<endl<<endl;
    }
    fin.close();
}

int main()
{
    int menuchoice;

    mainMenuSection:
    cout<<"\n\n\t#####\n\t#####HANGMAN#####\n\
\t#####2107103#####\n"<<endl;
    cout<<"1. Start New Game \n2. Check High Score \n3. Exit the game\n\nEnter your choice: ";
    tryAgain:
    cin>>menuchoice;
    if(menuchoice==1)
    {
        set<char> usedChars;
    startPlaying:
        cout<<"\n\nEnter Player Name: ";
        string name;
        cin>>name;
    playSameName:
        playerClass pl1;
        pl1.setPlayerName(name);
    }
}

```

```

parentPlayer *p1 = &pl1;
int wrongTurns=0,ifMatch=0,remainingLife=5,randomremainingLife;
string secretWordGame, wordClueGame, userWordGame;
secretWordClass swc1;
secretWordGame=swc1.generateSecretWord(&randomremainingLife);
wordClueGame=swc1.generateWordClue(randomremainingLife);
userWordGame=swc1.generateUserWord();
firstShow(userWordGame,wordClueGame);

char userInput;

while(remainingLife!=0)
{

    int countdownSeconds = 120;
    long long int zero=0;
    long long int *timeGone=&zero;

    auto startTime = chrono::high_resolution_clock::now();
    bool stopTimer=false;

    while (true)
    {

        if (*timeGone >= countdownSeconds && remainingLife!=0)
        {
            goto playAgainSection;
            countdownSeconds=0;
            stopTimer=true;
        }
        else if(wrongTurns==6 || remainingLife==0)
        {
            countdownSeconds=0;
            stopTimer=true;

            break;
        }

        cout<<"Enter a letter: ";
        cin>>userInput;
        auto elapsedTime =
chrono::duration_cast<chrono::seconds>(chrono::high_resolution_clock::now() -
startTime).count();

        timeGone=&elapsedTime;

```



```

checkInput(userInput,&wrongTurns,secretWordGame,userWordGame,wordClueGame,ifMatch,
p1,&remainingLife,usedChars,timeGone);
    this_thread::sleep_for(chrono::seconds(1));

    }
    usedChars.clear();
playAgainSection:
    char playAgain;
    cout<<"\n\nWant to play again?(y/n): ";
    cin>>playAgain;
    if(playAgain=='y' || playAgain=='Y')
    {
        char playerChange;
        cout<<"\nChange Player?(y/n): ";
        cin>>playerChange;
        if(playerChange=='y' || playerChange=='Y')
        {
            goto startPlaying;
        }
        else
        {
            goto playSameName;
        }
    }
    else
    {
        goto mainMenuSection;
    }
}
}
else if(menuchoice==2)
{
    showHighScore();
    goto mainMenuSection;
}
else if(menuchoice==3)
{
    cout<<"\n\nSee you again, Smart kiddo!"<<endl;
}
else
{
    cout<<"\nWrong input. Enter again: ";
    goto tryAgain;
}
return 0;

}

```

## Outputs


```
-----  
|  HANGMAN  |  
-----2107103-----
```

1. Start New Game
2. Check High Score
3. Exit the game

Enter your choice: 1

Enter Player Name: Ronaldo

```
=====
```



```
=====
```

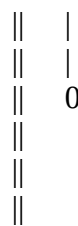
Clue: A musical instrument with wooden bars.

Timeline: \_\_\_\_\_

Time stars now... You have only 120seconds.

Enter a letter: a

```
=====
```



```
=====
```

Clue: A musical instrument with wooden bars.

Timeline: \_\_\_\_\_

Used Letters: A

Current Score: 84

Time remaining: 114

Enter a letter: w

=====

```
||  |
||  |
||  0
||  /
||
||
```

=====

Clue: A musical instrument with wooden bars.

Timeline: \_\_\_\_\_

Used Letters: A W

Current Score: 68

Time remaining: 111

Enter a letter: e

=====

```
||  |
||  |
||  0
||  /
||
||
```

=====

Clue: A musical instrument with wooden bars.

Timeline: \_\_\_\_\_E

Used Letters: A E W

Current Score: 68

Time remaining: 109

Enter a letter: l

=====

```
||  |
||  |
||  0
||  /
||
||
```

=====

Clue: A musical instrument with wooden bars.

Timeline: \_\_L\_\_\_\_E

Used Letters: A E L W

Current Score: 68

Time remaining: 107

Enter a letter: t

=====

|| |  
|| |  
|| 0  
|| /\n  
||  
||

=====

Clue: A musical instrument with wooden bars.  
Timeline: \_\_L\_\_\_\_E  
Used Letters: A E L T W  
Current Score: 52  
Time remaining: 105

Enter a letter: s

=====

|| |  
|| |  
|| 0  
|| /\n  
||  
||

=====

Clue: A musical instrument with wooden bars.  
Timeline: \_\_L\_\_\_\_E  
Used Letters: A E L S T W  
Current Score: 36  
Time remaining: 103

Enter a letter: s

=====

|| |  
|| |  
|| 0  
|| /\n  
||  
||

=====

Clue: A musical instrument with wooden bars.  
Timeline: \_\_L\_\_\_\_E  
Used Letters: A E L S T W  
Current Score: 36  
Time remaining: 101

Enter a letter: v

=====

|| |

```
||  |
||  0
||  /\
||  /
||
```

=====

Clue: A musical instrument with wooden bars.

Timeline: \_\_L\_\_\_\_E

Used Letters: A E L S T V W

Current Score: 20

Time remaining: 99

Enter a letter: r

=====

```
||  |
||  |
||  0
||  /\
||  /\
||
```

=====

Clue: A musical instrument with wooden bars.

Timeline: \_\_L\_\_\_\_E

Used Letters: A E L R S T V W

Current Score: 0

Time remaining: 97

You are dead, Ronaldo! The word was: XYLOPHONE

Want to play again?(y/n): y

Change Player?(y/n): y

Enter Player Name: Messi

=====

```
||  |
||  |
||
||
||
||
```

=====

Clue: A luxurious boat often used for sailing.

Timeline: \_\_\_\_\_

Time stars now... You have only 120seconds.

Enter a letter: y

```
=====
||  |
||  |
||
||
||
||
=====
```

Clue: A luxurious boat often used for sailing.

Timeline: Y\_\_\_\_\_

Used Letters: Y

Current Score: 100

Time remaining: 115

Enter a letter: a

```
=====
||  |
||  |
||
||
||
||
=====
```

Clue: A luxurious boat often used for sailing.

Timeline: YA\_\_\_\_\_

Used Letters: A Y

Current Score: 100

Time remaining: 114

Enter a letter: t

```
=====
||  |
||  |
||
||
||
||
=====
```

Clue: A luxurious boat often used for sailing.

Timeline: YA\_\_T\_\_\_\_\_

Used Letters: A T Y

Current Score: 100

Time remaining: 112

Enter a letter: h

=====

|| |  
|| |  
||  
||  
||  
||

=====

Clue: A luxurious boat often used for sailing.

Timeline: YA\_HT

Used Letters: A H T Y

Current Score: 100

Time remaining: 110

Enter a letter: c

=====

|| |  
|| |  
||  
||  
||  
||

=====

Clue: A luxurious boat often used for sailing.

Timeline: YACHT

Used Letters: A C H T Y

Current Score: 100

Time remaining: 109

Congo, Messi! Your Score is: 100

Want to play again?(y/n): n

-----  
| HANGMAN |  
-----2107103-----

1. Start New Game
2. Check High Score
3. Exit the game

Enter your choice: 2

Latest Highest Scorer: Messi

Highest Score: 100

```
-----  
|  HANGMAN  |  
-----2107103-----
```

1. Start New Game
2. Check High Score
3. Exit the game

Enter your choice: 3

See you again, Smart kiddo!

Process returned 0 (0x0) execution time : 56.712 s  
Press any key to continue.

## **Code Explanation**

### **Classes:**

1. parentPlayer and playerClass Classes:

parentPlayer is an abstract class that defines the blueprint for player objects. It includes pure virtual functions setPlayerName, setPlayerScore, and getPlayerName, which are overridden by the concrete playerClass. playerClass is a derived class from parentPlayer, playerClass represents individual players. It implements the pure virtual functions and manages every player's name and score.

2. secretWordClass Class:

secretWordClass manages secret words and their clues. It has functions to generate a secret word, generate a word clue, and create a user word with underscores to represent the word's letters.

## **Game Logic and Functions**

1. checkInput(...) Function:

checkInput function handles the core game logic. It takes input entered by the user, checks if it matches letters in the secret word, updates the game state, and displays



relevant information. It also manages incorrect guesses and the Hangman figure for incorrect guess.

## 2. firstShow( ) Function:

This function displays the initial game interface, including the Hangman figure, word clue, and user word with underscores. It also sets the timer for the game.

## 3. showHighScore( ) Function:

showHighScore function reads high scores from a file (scores.txt) and displays the latest highest scorer's name and their score.

## **Time Management:**

Timer Implementation: The code integrates a timer into the gameplay using the 'chrono' library. It allows each player to play withing a fixed time limit of 120 seconds to make their guesses. This is implemented through time-tracking variable timeGone.

## **Game Flow**

**Player Initialization:** When a player enters their name, a playerClass object is created to represent them. The default score is set to 100.

Secret Word Selection: A random secret word is selected from a predefined list made using string array, and the corresponding word clue is provided.

**Gameplay:** Players take turns by guessing only one letter at a time. Correct guesses reveal letters through the userWord variable, while keeping the score unchanged. Incorrect guesses decrement the player's score and display the Hangman figure as wrong guesses accumulate. When the Hangman figure is fully loaded the game ends showing the secret word.

## **Discussions**

In this project, a commendable application of Object-Oriented Programming (OOP) principles is observed. The code effectively employs a range of fundamental OOP concepts, including classes, objects, inheritance, and polymorphism. These principles play a crucial role in structuring the codebase, facilitating data encapsulation, and ensuring the efficient execution of the game. Also friend functions which manages access to private data elements. However, there are a couple of OOP concepts that could have further elevated the program's quality. Operator overloading, for instance, could have enhanced the code's expressiveness and readability. Additionally, the introduction of friend classes could have provided greater flexibility in managing class relationships. Though the utilization of various OOP concepts has yielded a satisfying and functional outcome

## **Conclusion**

Through the use of abstract classes, inheritance, and polymorphism, the code effectively manages player-related data and behaviors, ensuring a consistent and extensible interface for player objects. The OOP design of the Hangman game not only promotes clean code architecture but also provides a solid foundation for future enhancements and the addition of new features. It shows the significance of OOP principles in creating well-structured and maintainable C++ applications.