Abstract Base Class & Pure Virtual Functions in C++ | C++ Tutorials for Beginners #58

In this tutorial, we will discuss abstract base class and pure virtual functions in C++

Pure Virtual Functions in C++

Pure virtual function is a function that doesn't perform any operation and the function is declared by assigning the value 0 to it. Pure virtual functions are declared in abstract classes.

Abstract Base Class in C++

Abstract base class is a class that has at least one pure virtual function in its body. The classes which are inheriting the base class must need to override the virtual function of the abstract class otherwise compiler will throw an error.
To demonstrate the concept of abstract class and pure virtual function an example program is shown below.

```cpp
class CWH{
    protected:
        string title;
        float rating;
    public:
        CWH(string s, float r){
            title =  s;
            rating = r;
        }
        virtual void display()=0;
};
```

Copy

Code Snippet 1: Code with Harry Class

As shown in code snippet 1,

1. We created a class "CHW" which contains protected data members "title" which has "string" data type and "rating" which has "float" data type.
2. The class "CWH" has a parameterized constructor which takes two parameters "s" and "r" and assign their values to the data members "title" and "rating"
3. The class "CHW" has a pure virtual function void "display" which is declared by 0. The main thing to note here is that as the "display" function is a pure virtual function it is compulsory to redefine it in the derived classes.

```
class CWHVideo: public CWH
{
    float videoLength;
    public:
        CWHVideo(string s, float r, float vl): CWH(s, r){
            videoLength = vl;
        }
        void display(){
            cout<<"This is an amazing video with title "<<title<<endl;
            cout<<"Ratings: "<<rating<<" out of 5 stars"<<endl;
            cout<<"Length of this video is: "<<videoLength<<" 
minutes"<<endl;
        }
};
```

Copy

Code Snippet 2: Code with Harry Video Class

As shown in code snippet 2,

1. We created a class “CHWVideo” which is inheriting “CWH” class and contains private data members “videoLength” which has “float” data type.
2. The class “CWHVideo” has a parameterized constructor which takes three parameters “s”, “r” and “vl”. The constructor of the base class is called in the derived class and the values of the variables “s” and “r” are passed to it. The value of the parameter “vl” will be assigned to the data members “videoLength”
3. The class “CHWVideo” has a function void “display” which will print the values of the data members “title”, “rating” and “videoLength”

```
class CWHText: public CWH
{
    int words;
    public:
        CWHText(string s, float r, int wc): CWH(s, r){
            words = wc;
        }
     void display(){
      cout<<"This is an amazing text tutorial with title "<<title<<endl;
      cout<<"Ratings of this text tutorial: "<<rating<<" out of 5 stars"<<endl;
      cout<<"No of words in this text tutorial is: "<<words<<" words"<<endl;
         }
};
```

Copy

Code Snippet 3: Code with Harry Text Class

As shown in code snippet 3,

1. We created a class “CHWText” which is inheriting “CWH” class and contains private data members “words” which has “int” data type.
2. The class “CWHText” has a parameterized constructor which takes three parameters “s”, “r” and “wc”. The constructor of the base class is called in the derived class and the values of the variables “s” and “r” are passed to it. The value of the parameter “wc” will be assigned to the data members “words”
3. The class “CHWText” has a function void “display” which will print the values of the data members “title”, “rating” and “words”

```
int main(){
    string title;
    float rating, vlen;
    int words;

    // for Code With Harry Video
    title = "Django tutorial";
    vlen = 4.56;
    rating = 4.89;
    CWHVideo djVideo(title, rating, vlen);

    // for Code With Harry Text
    title = "Django tutorial Text";
    words = 433;
    rating = 4.19;
    CWHText djText(title, rating, words);

    CWH* tuts[2];
    tuts[0] = &djVideo;
    tuts[1] = &djText;

    tuts[0]->display();
    tuts[1]->display();

    return 0;
}
```

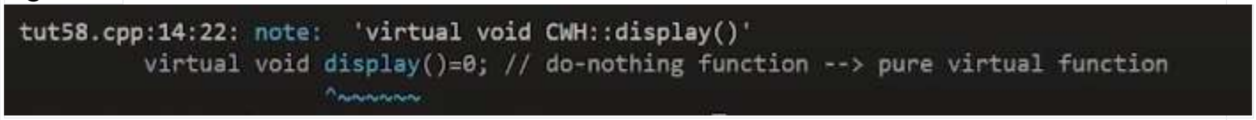
Copy

Code Snippet 4: Main Program

As shown in code snippet 4,

1. We created a string variable “title”, float variables “rating”, “vlen” and integer variable “words”
2. For the code with harry video class we have assigned “Django tutorial” to the string “title”, “4.56” to the float “vlen” and “4.89” to the float “rating”.
3. An object “djVideo” is created of the data type “CWHVideo” and the variables “title”, “rating” and “vlen” are passed to it.
4. For the code with harry text class we have assigned “Django tutorial text” to the string “title”, “433” to the integer “words” and “4.19” to the float “rating”.
5. An object “djText” is created of the data type “CWHText” and the variables “title”, “rating” and “words” are passed to it.
6. Two pointers array “tuts” is created of the “CWH” type
7. The address of the “djVideo” is assigned to “tuts[0]” and the address of the “djText” is assigned to “tuts[1]”
8. The function “display” is called using pointers “tuts[0]” and “tuts[1]”

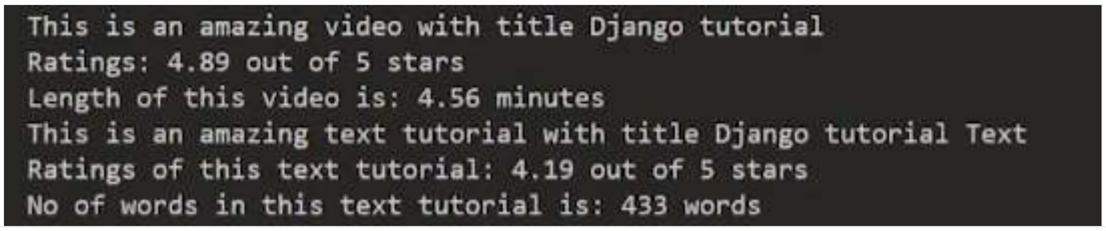
The main thing to note here is that if we don’t override the pure virtual function in the derived class the compiler will throw an error as shown in figure 1.



```
tut58.cpp:14:22: note:   'virtual void CWH::display()'
         virtual void display()=0; // do-nothing function --> pure virtual function
                      ^~~~~~~
```

Figure 1: Program Error

The output of the following program is shown in figure 2



```
This is an amazing video with title Django tutorial
Ratings: 4.89 out of 5 stars
Length of this video is: 4.56 minutes
This is an amazing text tutorial with title Django tutorial Text
Ratings of this text tutorial: 4.19 out of 5 stars
No of words in this text tutorial is: 433 words
```