Digital Clock in C++

Project overview

Conditions and loops are two important C++ concepts that we will practice when building the digital clock.

We'll use the time () function to get the local time.

To initialize our variables, we will use the tm structure, which contains data and time characteristics.

Let's break down the digital clock program into smaller steps to make it easier to understand and complete.

The following actions must be implemented:

- Use the time () method to determine the current system time.
- Initialize the tm structure with the hours, minutes, and seconds declarations.
- Show the current time on a digital clock using a while loop.
- Increase the hours, minutes, and seconds variables depending on the current situation and the input.
- Add a delay and then delete content from the screen.

Obtaining the current system time

We use the following procedure to obtain the current time:

- Use the time library's time () method in C++. It provides an object of type time with the current time as a value.
- Use the local time () method to convert a time to a tm identifier. An identifier is a name used to refer to a class of objects.

- Declare a Timer ptr type pointer to hold the value returned by the local time() function.
- The tm type allows us to manipulate time using characteristics such as tm sec, tm min, tm hour, and so on.

Utilizing struct attributes

The arrow operator may be used to retrieve the properties of timeptr.

Set the time sec property to the value of the sec variable that you declared.

Initialize the variable min with the tm min attribute before declaring another one with the same name.

Use the tm hour property to set the hours variable to zero. Then, declare an AM/PM timestr variable.

The code below stores the local time in variables using pointers.
The if condition is used to change the local time to the 12-hour clock format.

```
//getting values of seconds, minutes and hours

   struct tm* ct=localtime(&total_seconds);



   seconds=ct->tm_sec;

   minutes=ct->tm_min;

   hours=ct->tm_hour;
```

INCREMENT THE TIME:-

- Increment the `sec` variable on every iteration of the while loop.
- Once the `sec` value reaches `60`, increment the `min` variable by `one`. Reset the `sec` back to one.
- In the same way, when the `min` reaches `60`, increment hours by `one` and reset the `min` variable to `0`.
- Set the hours to `00` when it reaches 24. This is because the standard time in the 24-hour system ranges from one to twenty-four.

```
while (true)
    {
        // This increases the seconds
        sec++;
        if (seconds >= 60)
        {
            seconds = 1;
            minutes++;
        }
        // This increases the minutes
        if (minutes >= 60)
        {
```

```
            minutes = 0;
            hrs++;
        }
        // This increases the hours
        if (hrs >= 24)
        {
            hrs = 00;
        }
    }
```

After incrementing, the last step is to add a delay and clear the screen simultaneously.

To achieve this functionality, we will use the following steps:

- Use `system(cls)` to clear the `view`.
- We will add a `1000` ms delay using the `sleep()` function.

Displaying the digital clock:

```
  system("cls");
  system("color 1f");
      // This output the message "The digital time is :"
      cout<<endl;
    cout<<endl;
    cout<<endl;
    cout<<endl;
    cout<<endl;
    cout<<endl;
    cout<<endl;
    cout<<endl;
    cout << "                                DIGITAL TIME";
    cout<<endl;
    cout<<"
**************************"<<endl;
    cout<<endl;
    cout<<endl;
    cout << "                                " << hrs << " : " << minutes << " : " <<
seconds << " " << endl;
    cout<<endl;
```

```
cout<<endl;

cout<<"                          **************************";
```