# C++ Datum und Uhrzeit

Die C++ standard-Bibliothek nicht eine richtige Datum-Typ. C++ erbt die Strukturen und Funktionen für Datum und Zeit Bearbeitung von C. An access Datum und Zeit verwandten Funktionen und Strukturen, würden Sie brauchen, um include <ctime> header-Datei in Ihrem C++ - Programm.

Es gibt vier Zeit-Typen: **clock_t, time_t, size_t** , und **tm** . Die Typen - clock_t, size_t und time_t in der Lage sind, repräsentieren die system-Zeit und Datum als eine Art von integer.

Die Struktur-Typ **tm** hält das Datum und die Uhrzeit in der form der C-Struktur mit den folgenden Elementen

```
struct tm {
   int tm_sec;   // seconds of minutes from 0 to 61
   int tm_min;   // minutes of hour from 0 to 59
   int tm_hour;  // hours of day from 0 to 24
   int tm_mday;  // day of month from 1 to 31
   int tm_mon;   // month of year from 0 to 11
   int tm_year;  // year since 1900
   int tm_wday;  // days since sunday
   int tm_yday;  // days since January 1st
   int tm_isdst; // hours of daylight savings time
}
```

Im folgenden sind die wichtigsten Funktionen, die wir verwenden, während der Arbeit mit Datum und Zeit, die in C oder C++. All diese Funktionen sind Teil von standard-C-und C++ - Bibliothek, und Sie können überprüfen Ihre detail mit Verweis auf C++ - standard-Bibliothek, die unten gegeben werden.

| Sr. No | Funktion & Zweck |
|---|---|
| 1 | **time_t time(time_t *time);**<br><br>This returns the current calendar time of the system in number of seconds elapsed since January 1, 1970. If the system has no time, .1 is returned. |
| 2 | **char *ctime(const time_t *time);**<br><br>This returns a pointer to a string of the form *day month year hours:minutes:seconds year\n\0*. |
| 3 | **struct tm *localtime(const time_t *time);**<br><br>This returns a pointer to the **tm** structure representing local time. |
| 4 | **clock_t clock(void);**<br><br>This returns a value that approximates the amount of time the calling program has been running. A value of .1 is returned if the time is not available. |
| 5 | **char * asctime ( const struct tm * time );** |

| | |
|---|---|
| | This returns a pointer to a string that contains the information stored in the structure pointed to by time converted into the form: day month date hours:minutes:seconds year\n\0 |
| 6 | **struct tm *gmtime(const time_t *time);** This returns a pointer to the time in the form of a tm structure. The time is represented in Coordinated Universal Time (UTC), which is essentially Greenwich Mean Time (GMT). |
| 7 | **time_t mktime(struct tm *time);** This returns the calendar-time equivalent of the time found in the structure pointed to by time. |
| 8 | **double difftime ( time_t time2, time_t time1 );** This function calculates the difference in seconds between time1 and time2. |
| 9 | **size_t strftime();** This function can be used to format date and time in a specific format. |

## Current Date and Time

Suppose you want to retrieve the current system date and time, either as a local time or as a Coordinated Universal Time (UTC). Following is the example to achieve the same −

Live Demo

```
#include <iostream>
#include <ctime>

using namespace std;

int main() {
   // current date/time based on current system
   time_t now = time(0);

   // convert now to string form
   char* dt = ctime(&now);

   cout << "The local date and time is: " << dt << endl;

   // convert now to tm struct for UTC
   tm *gmtm = gmtime(&now);
   dt = asctime(gmtm);
   cout << "The UTC date and time is:"<< dt << endl;
}
```

When the above code is compiled and executed, it produces the following result −

```
The local date and time is: Sat Jan  8 20:07:41 2011

The UTC date and time is:Sun Jan  9 03:07:41 2011
```

## Format Time using struct tm

The **tm** structure is very important while working with date and time in either C or C++. This structure holds the date and time in the form of a C structure as mentioned above. Most of the time related functions makes use of tm structure. Following is an example which makes use of various date and time related functions and tm structure −

While using structure in this chapter, I'm making an assumption that you have basic understanding on C structure and how to access structure members using arrow -> operator.

Live-Demo

```
#include <iostream>
#include <ctime>

using namespace std;

int main() {
   // current date/time based on current system
   time_t now = time(0);

   cout << "Number of sec since January 1,1970 is:: " << now << endl;

   tm *ltm = localtime(&now);

   // print various components of tm structure.
   cout << "Year:" << 1900 + ltm->tm_year<<endl;
   cout << "Month: "<< 1 + ltm->tm_mon<< endl;
   cout << "Day: "<< ltm->tm_mday << endl;
   cout << "Time: "<< 5+ltm->tm_hour << ":";
   cout << 30+ltm->tm_min << ":";
   cout << ltm->tm_sec << endl;
}
```

Wenn der obige code kompiliert und ausgeführt wird, erzeugt es Folgendes Ergebnis −

```
Number of sec since January 1,1970 is:: 1588485717
Year:2020
Month: 5
Day: 3
Time: 11:31:57
```

## 1. Beispiel

```
#include <iostream>
#include <ctime>

using namespace std;

int main() {
   // current date/time based on current system
   time_t now = time(0);

   // convert now to string form
   char* dt = ctime(&now);

   cout << "The local date and time is: " << dt << endl;

   // convert now to tm struct for UTC
   tm *gmtm = gmtime(&now);
   dt = asctime(gmtm);
   cout << "The UTC date and time is:"<< dt << endl;
}
```

## 2. Beispiel

```
#include <iostream>
#include <ctime>

using namespace std;

int main() {
   // current date/time based on current system
   time_t now = time(0);

   cout << "Number of sec since January 1,1970 is:: " << now << endl;

   tm *ltm = localtime(&now);

   // print various components of tm structure.
   cout << "Year:" << 1900 + ltm->tm_year<<endl;
   cout << "Month: "<< 1 + ltm->tm_mon<< endl;
   cout << "Day: "<<  ltm->tm_mday << endl;
   cout << "Time: "<< 5+ltm->tm_hour << ":";
   cout << 30+ltm->tm_min << ":";
   cout << ltm->tm_sec << endl;
}
```