C++ 引用 C++ 基本的输入输出

C++ 日期 & 时间

C++ 标准库没有提供所谓的日期类型。C++ 继承了 C 语言用于日期和时间操作的结构和函数。为了使用日期和时间相关的函数和结构，需要在 C++ 程序中引用 <ctime> 头文件。

有四个与时间相关的类型：**clock_t、time_t、size_t** 和 **tm**。类型 clock_t、size_t 和 time_t 能够把系统时间和日期表示为某种整数。

结构类型 **tm** 把日期和时间以 C 结构的形式保存，tm 结构的定义如下：

```
struct tm {
int tm_sec; // 秒，正常范围从 0 到 59，但允许至 61
int tm_min; // 分，范围从 0 到 59
int tm_hour; // 小时，范围从 0 到 23
int tm_mday; // 一月中的第几天，范围从 1 到 31
int tm_mon; // 月，范围从 0 到 11
int tm_year; // 自 1900 年起的年数
int tm_wday; // 一周中的第几天，范围从 0 到 6，从星期日算起
int tm_yday; // 一年中的第几天，范围从 0 到 365，从 1 月 1 日算起
int tm_isdst; // 夏令时
}
```

下面是 C/C++ 中关于日期和时间的重要函数。所有这些函数都是 C/C++ 标准库的组成部分，您可以在 C++ 标准库中查看一下各个函数的细节。

序号	函数 & 描述
1	time_t time(time_t *time); 该函数返回系统的当前日历时间，自 1970 年 1 月 1 日以来经过的秒数。如果系统没有时间，则返回 .1。
2	char *ctime(const time_t *time); 该返回一个表示当地时间的字符串指针，字符串形式 *day month year hours:minutes:seconds year\n\0*。
3	struct tm *localtime(const time_t *time); 该函数返回一个指向表示本地时间的 **tm** 结构的指针。
4	clock_t clock(void); 该函数返回程序执行起（一般为程序的开头），处理器时钟所使用的时间。如果时间不可用，则返回 .1。
5	char * asctime (const struct tm * time); 该函数返回一个指向字符串的指针，字符串包含了 time 所指向结构中存储的信息，返回形式为：day month date hours:minutes:seconds year\n\0。
6	struct tm *gmtime(const time_t *time); 该函数返回一个指向 time 的指针，time 为 tm 结构，用协调世界时（UTC）也被称为格林尼治标准时间（GMT）表

	示。
7	time_t mktime(struct tm *time); 该函数返回日历时间，相当于 time 所指向结构中存储的时间。
8	double difftime (time_t time2, time_t time1); 该函数返回 time1 和 time2 之间相差的秒数。
9	size_t strftime(); 该函数可用于格式化日期和时间为指定的格式。

当前日期和时间

下面的实例获取当前系统的日期和时间，包括本地时间和协调世界时（UTC）。

实例

```
#include <iostream>
#include <ctime>
using namespace std;
int main( )
{
// 基于当前系统的当前日期/时间
time_t now = time(0);
// 把 now 转换为字符串形式
char* dt = ctime(&now);
cout << "本地日期和时间: " << dt << endl;
// 把 now 转换为 tm 结构
tm *gmtm = gmtime(&now);
dt = asctime(gmtm);
cout << "UTC 日期和时间: "<< dt << endl;
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
本地日期和时间: Sat Jan  8 20:07:41 2011

UTC 日期和时间: Sun Jan  9 03:07:41 2011
```

使用结构 tm 格式化时间

tm 结构在 C/C++ 中处理日期和时间相关的操作时，显得尤为重要。tm 结构以 C 结构的形式保存日期和时间。大多数与时间相关的函数都使用了 tm 结构。下面的实例使用了 tm 结构和各种与日期和时间相关的函数。

在练习使用结构之前，需要对 C 结构有基本的了解，并懂得如何使用箭头 -> 运算符来访问结构成员。

实例

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

```
using namespace std;
int main( )
{
// 基于当前系统的当前日期/时间
time_t now = time(0);
cout << "1970 到目前经过秒数:" << now << endl;
tm *ltm = localtime(&now);
// 输出 tm 结构的各个组成部分
cout << "年: "<< 1900 + ltm->tm_year << endl;
cout << "月: "<< 1 + ltm->tm_mon<< endl;
cout << "日: "<< ltm->tm_mday << endl;
cout << "时间: "<< ltm->tm_hour << ":";
cout << ltm->tm_min << ":";
cout << ltm->tm_sec << endl;
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
1970 到目前时间:1503564157
年: 2017
月: 8
日: 24
时间: 16:42:37
```

← C++ 引用 C++ 基本的输入输出 →

2 篇笔记

写笔记

以 **20**-**-** **:**:** 格式输出当前日期：

```
#include <iostream>
#include <ctime>
#include <stdlib.h>
#include <stdio.h>

using namespace std;

string  Get_Current_Date();

int main( )
{
    // 将当前日期以 20** - ** - ** 格式输出
    cout << Get_Current_Date().c_str() << endl;

    getchar();
    return 0;
```

```
}

string  Get_Current_Date()
{
    time_t nowtime;
    nowtime = time(NULL); //获取日历时间
    char tmp[64];
    strftime(tmp,sizeof(tmp),"%Y-%m-%d %H:%M:%S",localtime(&nowtime));
    return tmp;
}
```

输出格式类似：

```
2018-09-19 09:00:58
```

飞羽 1年前 (2018-01-24)

vs2017 上使用安全方法:

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

using namespace std;

#define TIMESIZE 26

int main() {
    // 时间
    time_t now = time(0);
    char dt[TIMESIZE];
    errno_t err;
    err = ctime_s(dt, TIMESIZE, &now);
    cout << "local time: " << dt << endl;
    cout << "timestamp: " << now << endl;
    struct tm ltm;
    localtime_s(<m, &now);
    cout << "年: " << 1900 + ltm.tm_year << endl;
    cout << "月: " << 1 + ltm.tm_mon << endl;
    cout << "日: " << ltm.tm_mday << endl;
    cout << "时间: " << ltm.tm_hour << ":";
    cout << ltm.tm_min << ":";
    cout << ltm.tm_sec << endl;
    return 0;
}
```

jailman 3个月前 (12-19)