**Explicação completa do código da aula Exibindo Data e Hora de Seu Computador!**

**EXPLICAÇÃO DA AULA** Exibindo Data e Hora do seu Computador

Obs: atente-se a ERRATA do Código pois não temos mais horário de verão e não precisamos colocar 1 + em hora, minuto e segundo

A struct tm presente na referência <http://www.cplusplus.com/reference/ctime/tm/> possui algumas características importantes. Estas características definem como serão retornados os valores da Data e Da Hora

1. struct tm {
2. int tm\_sec; //representa os segundos de 0 a 59
3. int tm\_min; //representa os minutos de 0 a 59
4. int tm\_hour; //representa as horas de 0 a 24
5. int tm\_mday: //dia do mês de 1 a 31
6. int tm\_mon; //representa os meses do ano de 0 a 11
7. int tm\_year; //representa o ano a partir de 1900 logo retorna (Ano – 1900)
8. int tm\_wday; //dia da semana de 0 (domingo) até 6 (sábado)
9. int tm\_yday; // dia do ano de 1 a 365
10. int tm\_isdst; //indica horário de verão se for diferente de zero
11. };

Os dados da struct tm dão auto explicativos em relação ao seus dados

Este dados abaixo representam bem e de forma precisa seus respectivos valores

1. int tm\_sec; //representa os segundos de 0 a 59
2. int tm\_min; //representa os minutos de 0 a 59
3. int tm\_hour; //representa as horas de 0 a 24
4. int tm\_mday: //dia do mês de 1 a 31

Logo para estes dados NÃO PRECISAMOS ADICIONAR 1 +

1. std::cout << "\nHora: " << MinhaHora->tm\_hour << ":" << MinhaHora->tm\_min << ":" << MinhaHora->tm\_sec;
2. std::cout << "\nDia: " << MinhaHora->tm\_mday;

Contudo existem dados que precisam ser explicados

int tm\_mon; //representa os meses do ano de 0 a 11

int tm\_year; //representa o ano a partir de 1900 logo retorna (Ano – 1900)

int tm\_wday; //dia da semana de 0 (domingo) até 6 (sábado)

Observe que a variável que representa o mês int tm\_mon faz a representação a partir de zero por convenção e questões de implementação da linguagem C

Então a representação será a seguinte para os meses e seus respectivos números representados na struct tm na variável tm\_mon que vai de [0 a 11]

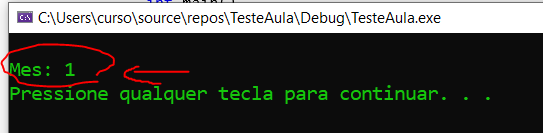
0 - Janeiro 1 – Fevereiro 2 – Março 3 - Abril  
4 – Maio 5 – Junho 6 – Julho 7 - Agosto  
8 – Setembro 9 – Outubro 10 – Novembro 11 – Dezembro

Então se você colocar na tela o mês atual irá colocar o mês anterior pois o índice dos meses da struct tm começa em zero

Veja

1. #define \_CRT\_SECURE\_NO\_WARNINGS
2. #include <iostream>
3. #include <ctime>
5. int main()
6. {
7. time\_t HoraAtual = time(0);
8. tm\* MinhaHora = localtime(&HoraAtual);
9. std::cout << "\nMes: " << MinhaHora->tm\_mon << "\n";
10. system("PAUSE");
11. return 0;
12. }

Estamos em fevereiro quando da edição deste documento, mas o que aparece para o usuário é:

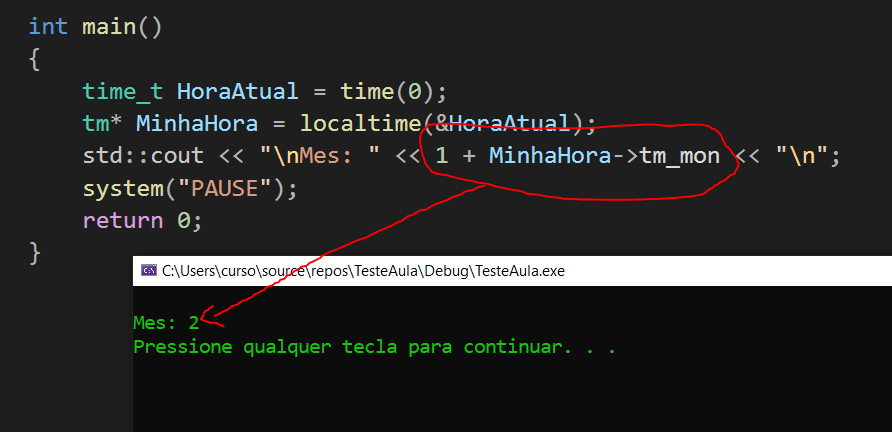


Como você é programador e sabe que 1 representa fevereiro, sabe que o resultado esta correto. Mas o usuário não sabe disso e então PRECISAMOS ADICIONAR UM AO RESULTADO DO MÊS PARA QUE NA TELA APAREÇA O MÊS CORRENTE

1 + MinhaHora->tm\_mon

1. #define \_CRT\_SECURE\_NO\_WARNINGS
2. #include <iostream>
3. #include <ctime>
5. int main()
6. {
7. time\_t HoraAtual = time(0);
8. tm\* MinhaHora = localtime(&HoraAtual);
9. std::cout << "\nMes: " << 1 + MinhaHora->tm\_mon << "\n";
10. system("PAUSE");
11. return 0;
12. }

Agora aparece corretamente

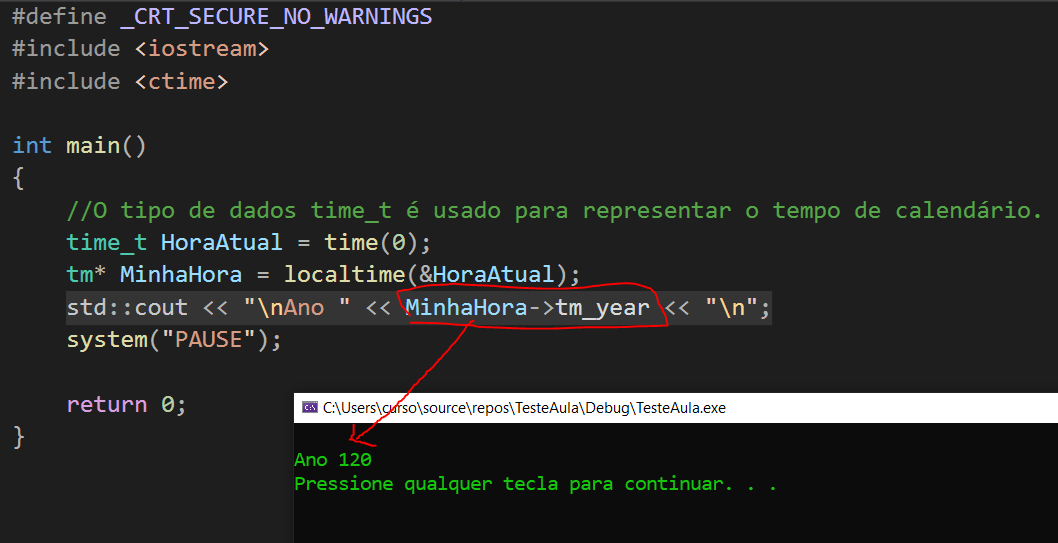


**EM RELAÇÃO AO ANO**

A variável int tm\_year retornará a quantidade de anos desde 1900. Isso é uma convenção criada deste 1970 na época em que da linguagem C do Unix POSIX. Mas esta convenção é seguida por outras arquiteturas(Windows, MacOS, etc)

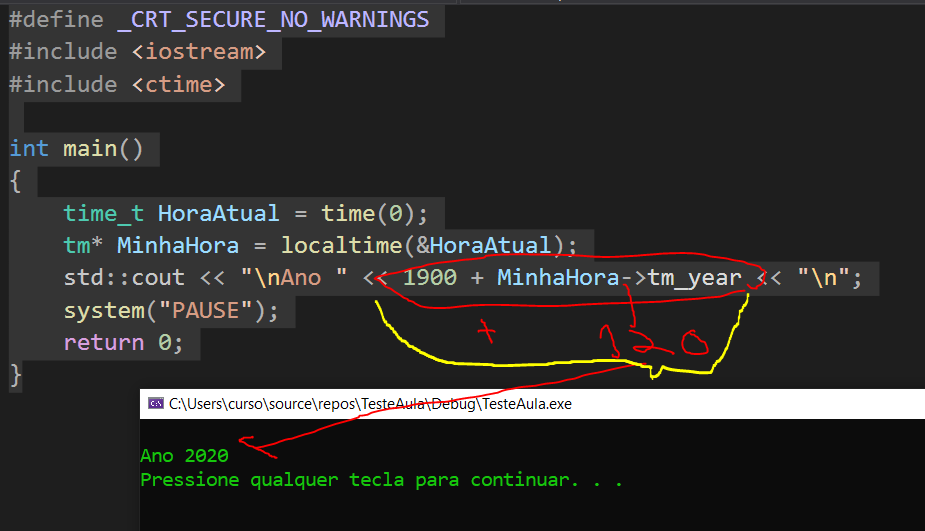
Então se colocarmos apenas std::cout << "\nAno " << MinhaHora->tm\_year << "\n"; o retorno será o número de anos decorridos deste 1900 assim será o mesmo que 2020 – 1900 e então dará 120 anos

1. #define \_CRT\_SECURE\_NO\_WARNINGS
2. #include <iostream>
3. #include <ctime>
5. int main()
6. {
7. //O tipo de dados time\_t é usado para representar o tempo de calendário.
8. time\_t HoraAtual = time(0);
9. tm\* MinhaHora = localtime(&HoraAtual);
10. std::cout << "\nAno " << MinhaHora->tm\_year << "\n";
11. system("PAUSE");
13. return 0;
14. }

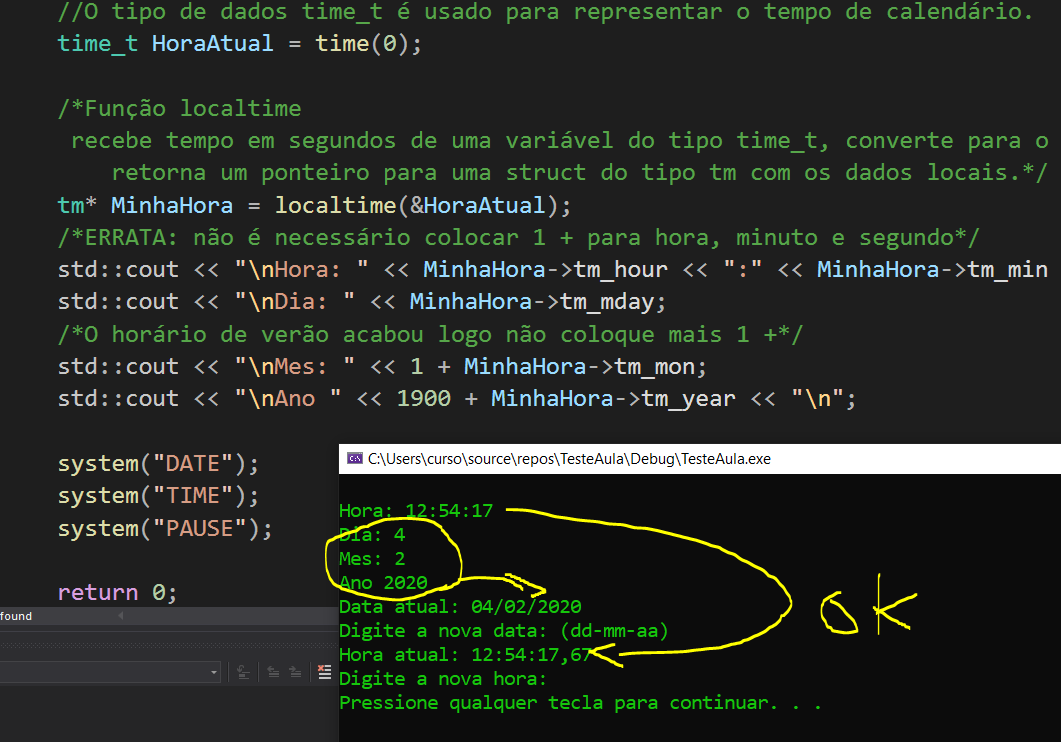


Mas não queremos este resultado! Queremos o ano atual! Logo para achar o ano atual basta somar 1900 a este resultado já que 1900 + 120 dará 2020 o ano atual que escrevi este texto. Observe o código abaixo:

1. #define \_CRT\_SECURE\_NO\_WARNINGS
2. #include <iostream>
3. #include <ctime>
5. int main()
6. {
7. time\_t HoraAtual = time(0);
8. tm\* MinhaHora = localtime(&HoraAtual);
9. std::cout << "\nAno " << 1900 + MinhaHora->tm\_year << "\n";
10. system("PAUSE");
11. return 0;
12. }



Logo coloque também antes de system(“PAUSE”) um outros systems com nome system(“Date”) e system(“TIME”). E pressionar ENTER duas vezes bem rápido Isso vai mostrar a data e hora atual no mesmo momento da execução do código.



Ai está o código final

1. #define \_CRT\_SECURE\_NO\_WARNINGS
2. #include <iostream>
3. #include <ctime>
5. int main()
6. {
7. //O tipo de dados time\_t é usado para representar o tempo de calendário.
8. time\_t HoraAtual = time(0);
10. /\*Função localtime
11. recebe tempo em segundos de uma variável do tipo time\_t, converte para o tempo local, armazena os dados na struct e retorna um ponteiro para uma struct do tipo tm com os dados locais.\*/
12. tm\* MinhaHora = localtime(&HoraAtual);
13. /\*ERRATA: não é necessário colocar 1 + para hora, minuto e segundo\*/
14. std::cout << "\nHora: " << MinhaHora->tm\_hour << ":" << MinhaHora->tm\_min << ":" << MinhaHora->tm\_sec;
15. std::cout << "\nDia: " << MinhaHora->tm\_mday;
16. /\*O horário de verão acabou logo não coloque mais 1 +\*/
17. std::cout << "\nMes: " << 1 + MinhaHora->tm\_mon;
18. std::cout << "\nAno " << 1900 + MinhaHora->tm\_year << "\n";
19. //pressione enter duas vezes bem rápido para ver os resultados sincronizados
20. system("DATE");
21. system("TIME");
22. system("PAUSE");
24. return 0;
25. }

A título de mais explicação no inicio do código temos uma declaração de variável

**time\_t** HoraAtual

O tipo de dados **time\_t** é usado para representar o tempo de calendário ou seja é o Tipo de horadata

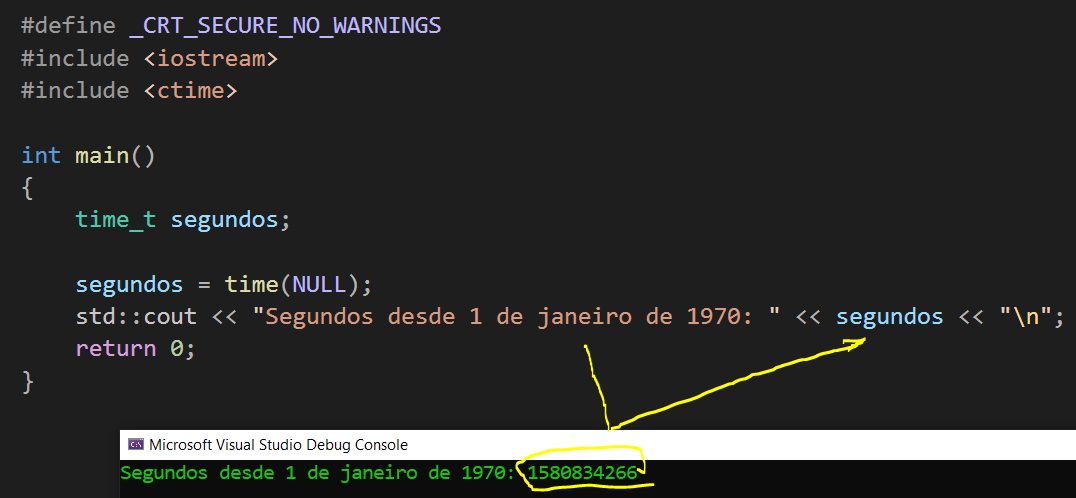
Este tipo é capaz de representar tempos, como aqueles retornados pela função time()

Em relação a função time() por razões históricas ela retorna um valor integral que representa o número de segundos decorridos desde as 00:00 horas, 1 de janeiro de 1970 UTC (ou seja, um registro de data e hora unix ).

Então se você executar o seguinte código

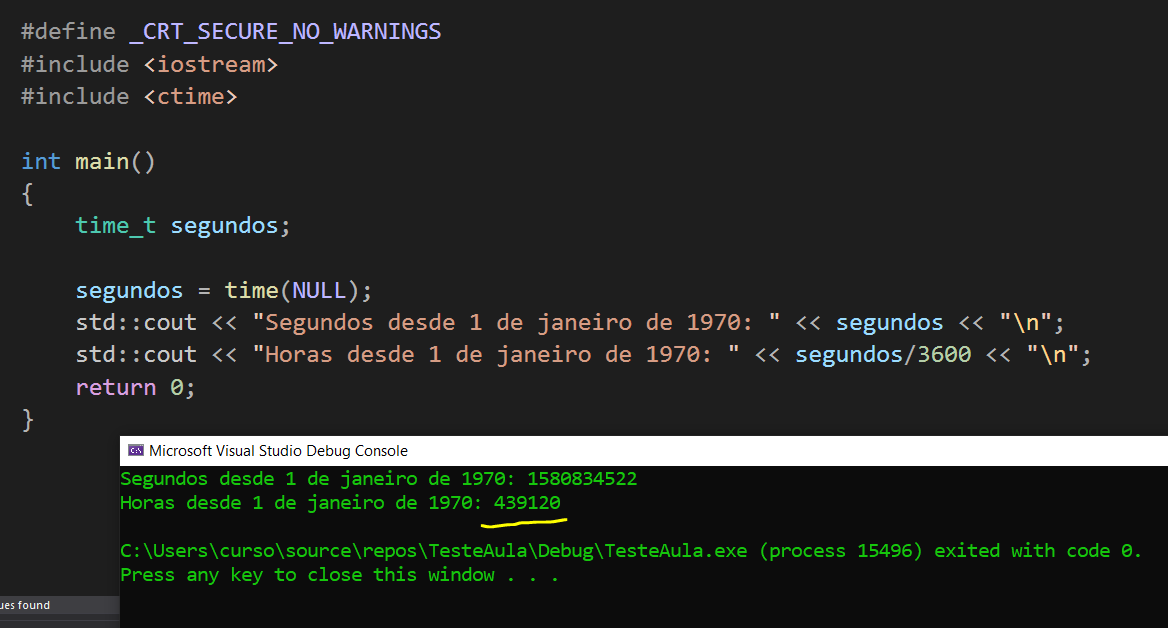
1. #define \_CRT\_SECURE\_NO\_WARNINGS
2. #include <iostream>
3. #include <ctime>
5. int main()
6. {
7. time\_t segundos;
9. segundos = time(NULL);
10. std::cout << "Segundos desde 1 de janeiro de 1970: " << segundos << "\n";
11. return 0;
12. }

Veja o resultado



Se quiser saber em horas basta dividir por 3600 pois uma hora tem 3600 segundos

1. #define \_CRT\_SECURE\_NO\_WARNINGS
2. #include <iostream>
3. #include <ctime>
5. int main()
6. {
7. time\_t segundos;
9. segundos = time(NULL);
10. std::cout << "Segundos desde 1 de janeiro de 1970: " << segundos << "\n";
11. std::cout << "Horas desde 1 de janeiro de 1970: " << segundos/3600 << "\n";
12. return 0;
13. }



Logo podemos obter estes segundos da seguinte forma time(0) ou time(NULL)

Por isso usamos time\_t HoraAtual = time(0);

A função localtime () pega um endereço de uma variável do tipo time\_t como argumento e retorna um ponteiro de uma estrutura do tipo tm. O valor retornado pela função localtime () é então a data e hora local.

Veja que local time retorna um ponteiro para do tipo struct tm já com os dados da hora e datas atuais

1. tm\* localtime(const time\_t\* time\_pretr);

Veja que enviamos para a função o endereço de HoraAtual para que ela monte a nossa variável MinhaHora com os dados da data e hora atuais seguindo o modelo da struct tm

tm\* MinhaHora = localtime(&HoraAtual);

EM RESEUMO

PRIMEIRO OBTEMOS O NÚMERO DE SEGUNDOS DECORRIDOS DESTE 1970

time\_t HoraAtual = time(0);

HoraAtual passa a ter este numero de segundo e esta variável é do tipo time\_t

Depois declaramos uma outra variável ponteiro do tipo struct tm de nome Minha hora e iremos atribuir o resultado da chamada da função localtime(&HoraAtual) para esta variável

Já que localtime recebe o endereço de uma variável do tipo time\_t ela irá receber &HoraAtual como parâmetro e irá devolver o endereço de uma struct tm criada de acordo com os dados dos segundos passados em HoraAtual... Ou seja localtime é capaz de criar uma struct com todos estes dados de hora, minuto, segundo, ano, mês, etc a partir desta informação.

tm\* MinhaHora = localtime(&HoraAtual);

e assim podemos no código usar MinhaHora para saber sobre a data e hora atuais

std::cout << "\nHora: " << MinhaHora->tm\_hour << ":" << MinhaHora->tm\_min << ":" << MinhaHora->tm\_sec;

std::cout << "\nDia: " << MinhaHora->tm\_mday;

/\*O horário de verão acabou logo não coloque mais 1 +\*/

std::cout << "\nMes: " << MinhaHora->tm\_mon;

std::cout << "\nAno " << 1900 + MinhaHora->tm\_year << "\n";