


```
for (i = -6 ; i <7 ; i++ )
{
    if (i==0) {printf ("\nnoon\n");};
    if (i==1) {printf ("noon\n\n"); };

    sinlat = sin( degtorad*lat ) ;

    // get the hour angle of the sun
    ii = (-1)*i ;
    j = 15 * (ii+corh) ;

    /* get the resulting hour line angle ~ atan(sin(lat)*tan(hr*15) */
    hlat = sinlat * tan(degtorad*j) ;

    /* get the hour line angle back to degrees */
    hla = radtodeg*( atan(hlat) ) ;

    printf ("Hour: %3i hour angle: %6.2f Angle: %6.2f \n", i, j, hla );
}

printf ( "afternoon hours last\n" );
printf ( " \n");
printf ( "**** END *** (enter any letter and return to quit)\n" ) ;

scanf ("%f", &ref); // a pause using any old variable
return 0;
}
```

"C" was the original language in the "C" series of languages. It was structured, in that the IF THEN program flow was strongly supported, and the GOTO was diminished.

As programming moved from sequential code towards event driven code, the concept of "objects" as opposed to simple variables or structures of variables came into play. And further, the concept came into existence of building code into the object to handle standard work that these objects might need done for them. These code segments, directly tied to the objects, were known as methods. This concept was used for many years in the IBM mainframe world where interrupt driven code was not new. However, as the UNIX world moved into more mainstream activity, the UNIX community considered these to be new concepts.

Be that as it may, the development of objects and their methods was formalized as a concept in the development of C++, a language derivative of "C". The object and its methods were abstracted further into classes.

A class was a definition of an object, or rather what it would be like. Thus a class was more of a dictionary, and the objects themselves were constructed from that dictionary. This was seen in the DeltaCAD definition of a human interaction window, and its subsequent generation or construction, and its subsequent use.

However, in a true object oriented system, the methods became inter-related. That interaction allowed simple events to cause the updating of many objects. One example is the Windows display of a folder of files, and how that display is updated automatically when in another window a file is added or dropped from that folder.

NOTE: January 2008: Microsoft provides a free Visual C++ Express system available at:-
<http://msdn.microsoft.com/vstudio/express/downloads/default.aspx>

These programs can be pasted directly into them if you use their Win32 Console selection, and they run. However, whereas these free systems from BLOODSHED are simple and unsophisticated, they produce executable files easily sent to others. Whereas the Visual C++ Express system is glossy and up to date, sending applications is nowhere near as simple. As an overview, this is shown on the next few pages.

March 31, 2007 ~ this may be distributed freely provided the web site credit
and this notice are retained. Tested on win32 XP as well as win64 Vista
revised April 26, 2009

PROGRAMMING IN VISUAL C++ EXPRESS (Microsoft)

Visual C++ Express is downloaded, first as a small installer, then as several very large files of stuff, which it then installs if the installer likes your system.

This is from Microsoft, and the registration process worked easily.

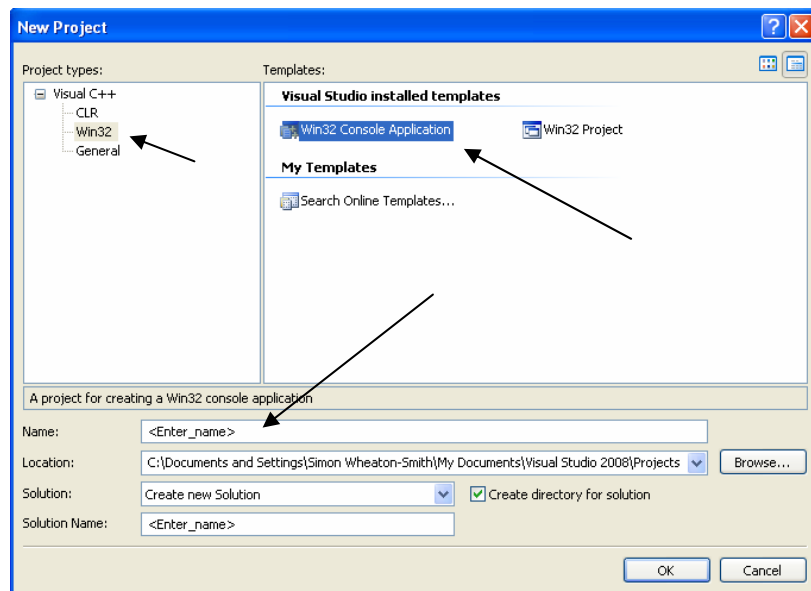
The url for the Visual Basic Express, and other light weight Express products is:-

<http://msdn.microsoft.com/vstudio/express/downloads/default.aspx>

And click on the Visual C++ 2008 (or whatever) DOWNLOAD panel, and proceed. Again, this is a free product. Remember, that while this is a free system, and while it is a lot more up to date than the other C compilers discussed by Illustrating Shadows, this Microsoft free product is not without its disadvantages. This Microsoft Visual C++ system, while free, does not allow for simple transmission of finished applications to others. Whereas the free C systems discussed elsewhere on the Illustrating Shadows web site and CD, while having less features, does allow for much less complicated transmission of completed applications.

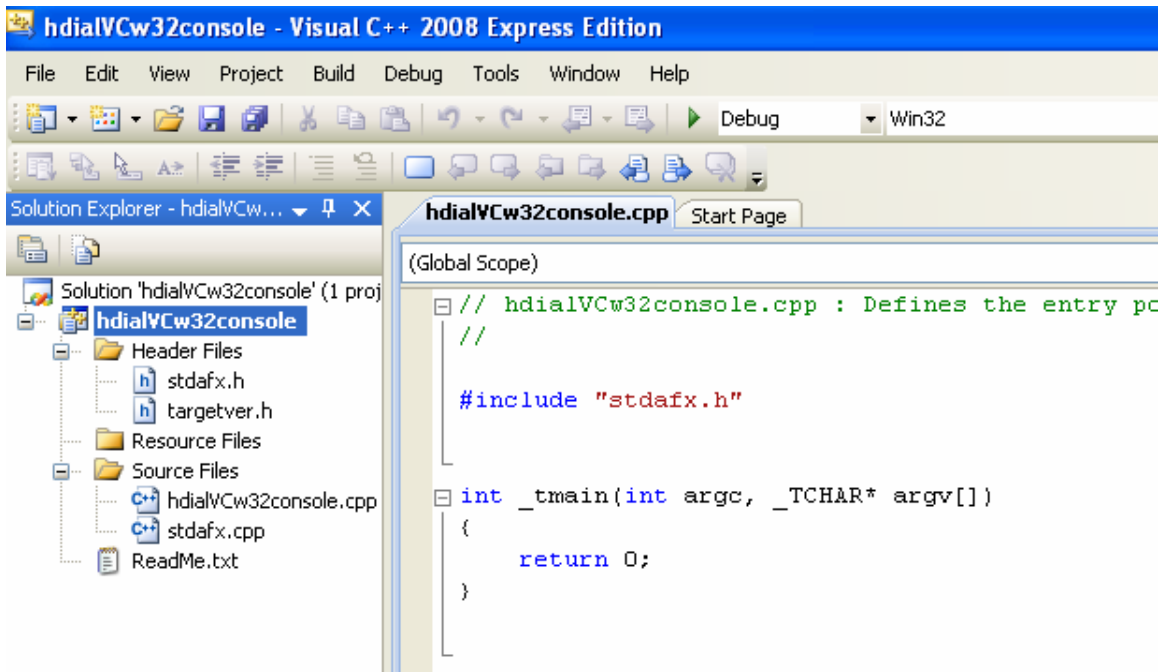
WINDOWS CONSOLE PROGRAM (The simplest)

The simplest programming is for a console program, such as run in a "DOS" window, one follows the same process, but instead of selecting Win32 Project, select Win32 Console Application, entering a name.

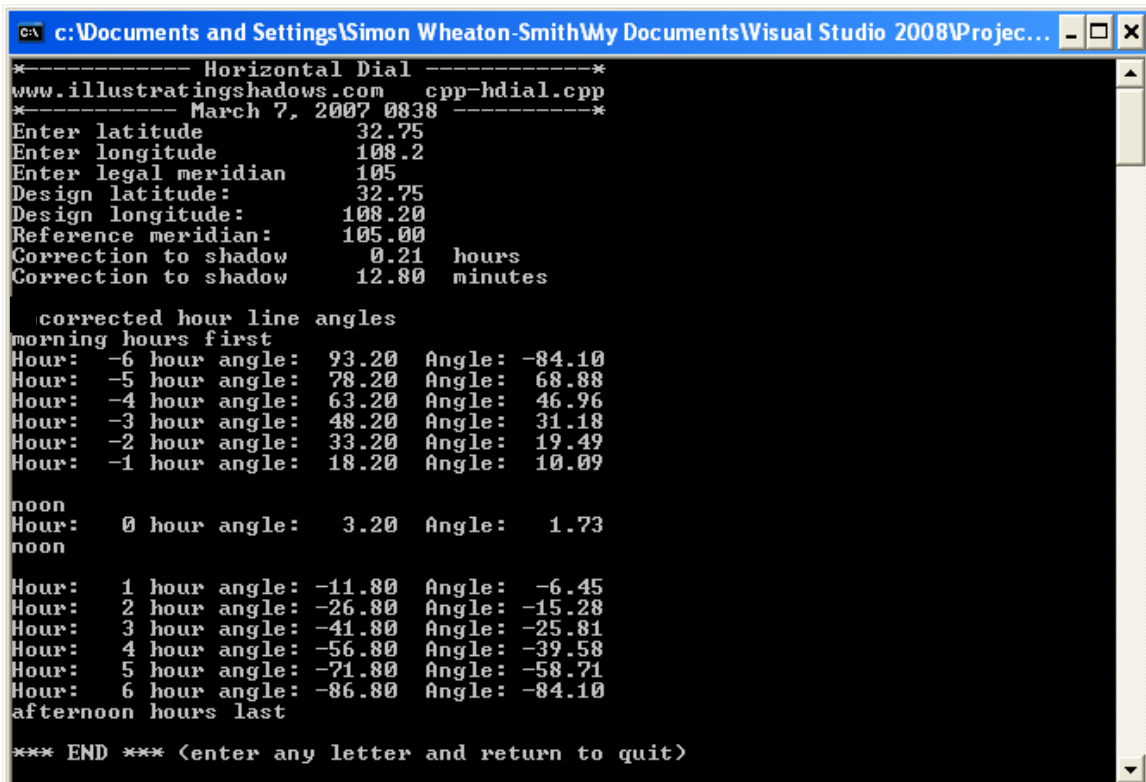


The next screen allows you to select Application Settings, and, as below, you select Console Application.

Then FINISH builds the project and generates sample code.



For the console program, running in a DOS window, the non windows C code may be pasted into the int _tmain section, the old main function being deleted, the #include and #define statements moved up to the template's #include, and the program tested with DEBUG, and that in turn triggers a build process, and upon completion, the program is run in a DOS window.



The source code would look like:-

```
// hdialVCw32console.cpp : Defines the entry point for the console
application.
//

#include "stdafx.h"
#include <stdio.h>
#include <math.h>

#define degtorad ((2 * 3.1416) / 360)
#define radtodeg (360 / (2 * 3.1416))

int _tmain(int argc, _TCHAR* argv[])
{
    int i , ii ;                // working integers
    float j;                    // working float

    float lat;                  // latitude and
    lat = 32.75 ;              // no good reason for this
    float sinlat;              // and sin of latitude
    float hlat ;               // hour line angle
    float lng;                  // longitude
    lng = 108.2 ;              // no good reason for this
    float ref;                  // reference meridian
    ref = 105 ;                // no good reason for this
    float corh;                 // correction for longitude
    float hla;                  // hour line angle

    printf ("*----- Horizontal Dial -----*\n");
    printf ("www.illustratingshadows.com  cpp-hdial.cpp\n");
    printf ("*----- March 7, 2007 0838 -----*\n");
    printf ("Enter latitude          ");
    scanf ("%f", &lat) ;
    printf ("Enter longitude          ");
    scanf ("%f", &lng);
    printf ("Enter legal meridian        ");
    scanf ("%f", &ref);

    printf ("Design latitude:          %7.2f \n", lat);
    printf ("Design longitude:           %7.2f \n", lng);
    printf ("Reference meridian:         %7.2f \n", ref);

    corh = (4*(lng-ref)) / 60 ;
    printf ("Correction to shadow    %7.2f  hours \n", corh);
    printf ("Correction to shadow    %7.2f  minutes \n", corh*60);
    printf ("\n");

    printf ("Corrected hour line angles \n" ) ;
    printf ("morning hours first\n");

    for (i = -6 ; i <7 ; i++ )
    {
        /* T E S T    V A L U E S
        *----- Horizontal Dial -----*
        www.illustratingshadows.com  cpp-hdial.cpp
        *----- March 7, 2007 0838 -----*
        Enter latitude          32.75

```

March 31, 2007 ~ this may be distributed freely provided the web site credit
and this notice are retained. Tested on win32 XP as well as win64 Vista
revised April 26, 2009

```
Enter longitude      108.2
Enter legal meridian 105
Design latitude:    32.75
Design longitude:   108.20
Reference meridian: 105.00
Correction to shadow 0.21 hours
Correction to shadow 12.80 minutes
```

Corrected hour line angles

morning hours first

```
Hour: -6 hour angle: 93.20 Angle: -84.10
Hour: -5 hour angle: 78.20 Angle: 68.88
Hour: -4 hour angle: 63.20 Angle: 46.96
Hour: -3 hour angle: 48.20 Angle: 31.18
Hour: -2 hour angle: 33.20 Angle: 19.49
Hour: -1 hour angle: 18.20 Angle: 10.09
```

noon

```
Hour: 0 hour angle: 3.20 Angle: 1.73
```

noon

```
Hour: 1 hour angle: -11.80 Angle: -6.45
Hour: 2 hour angle: -26.80 Angle: -15.28
Hour: 3 hour angle: -41.80 Angle: -25.81
Hour: 4 hour angle: -56.80 Angle: -39.58
Hour: 5 hour angle: -71.80 Angle: -58.71
Hour: 6 hour angle: -86.80 Angle: -84.10
```

afternoon hours last

```
*** END *** (enter any letter and return to quit)
```

```
*/
```

```
if (i==0) {printf ("\nnoon\n");};
if (i==1) {printf ("noon\n\n"); };
```

```
sinlat = sin( degtorad*lat ) ;
```

```
// get the hour angle of the sun
```

```
ii = (-1)*i ;
j = 15 * (ii+corh) ;
```

```
/* get the resulting hour line angle ~
```

```
atan(sin(lat)*tan(hr*15) */
hlat = sinlat *tan(degtorad*j) ;
```

```
/* get the hour line angle back to degrees */
```

```
hla = radtodeg*( atan(hlat) ) ;
```

```
printf ("Hour: %3i hour angle: %6.2f Angle: %6.2f \n", i,
j, hla );
}
```

```
printf ( "afternoon hours last\n" );
```

```
printf ( " \n");
```

```
printf ( "*** END *** (enter any letter and return to quit)\n" ) ;
```

```
scanf ("%f", &ref); // a pause using any old variable
```

```
return 0;
```

```
}
```