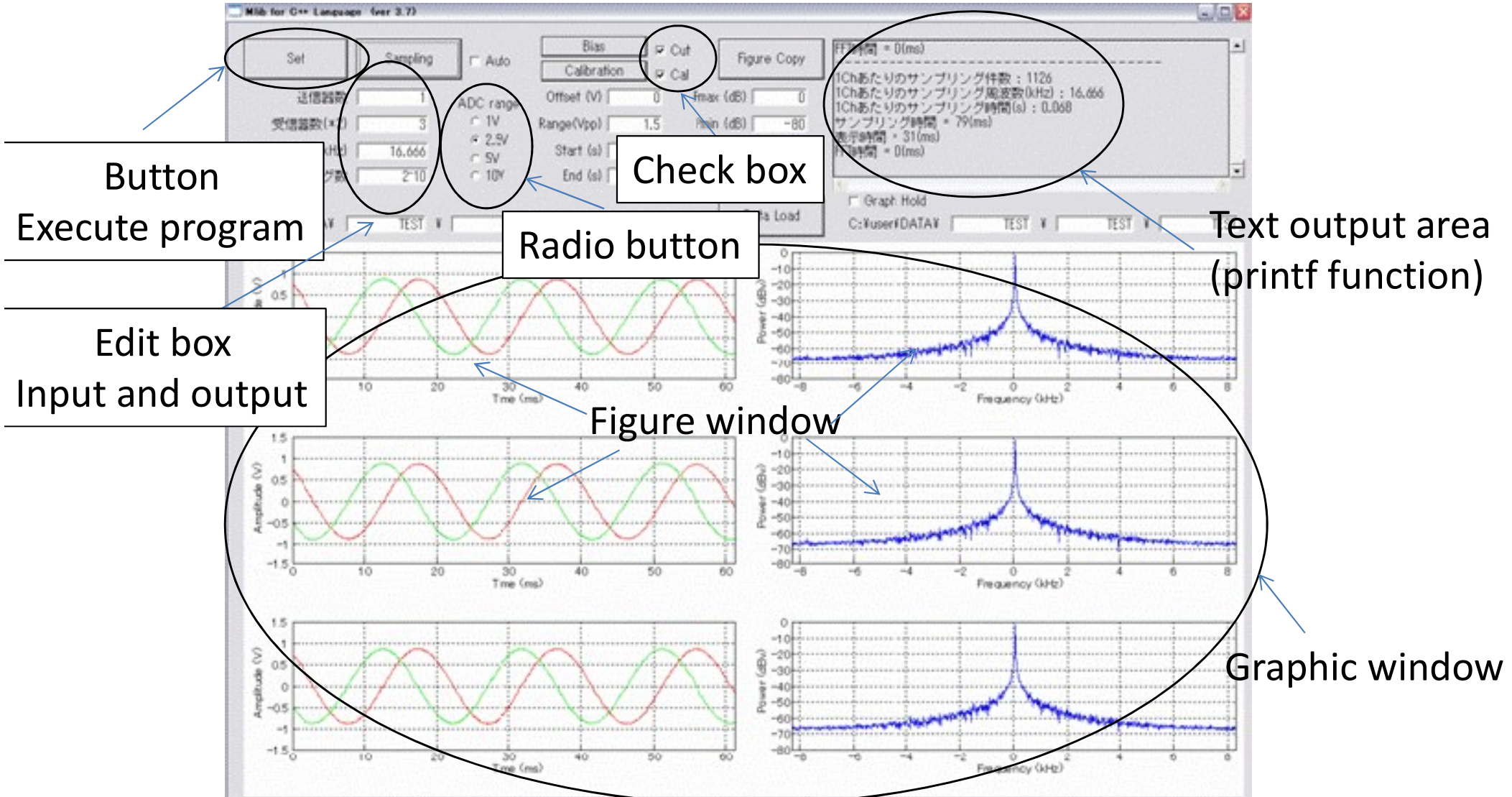
Contents

Ver. 1.0

What is mlib 3.8?

Mlib is GUI(Graphical User Interface) based data visualization library on MS Visual C++

1. Input and output of text or number through edit box
2. Checkbox or radio button for flag condition Setting
3. 1D drawing or 2D image of array data
4. Animation of objet with simple shape



Microsoft Visual Studio

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps.

Visual Studio supports many programming languages such as C++, Visual Basic.NET, C# etc. This lecture uses only C++ for Windows application.

The most basic edition of Visual Studio, the Community edition, is available free of charge. The Express edition is older version and is also free.

Visual Studio Community 2019 ← Up-to-date

Visual Studio Community 2017

Visual Studio Community 2015

free, Light size (you can choose programing language)

Visual Studio Express 2017 ← finished

⋮

Visual Studio Express 2010

free, Heavy size(many programing language is install)

Whichever you can use.
Community 2019 is recommended.

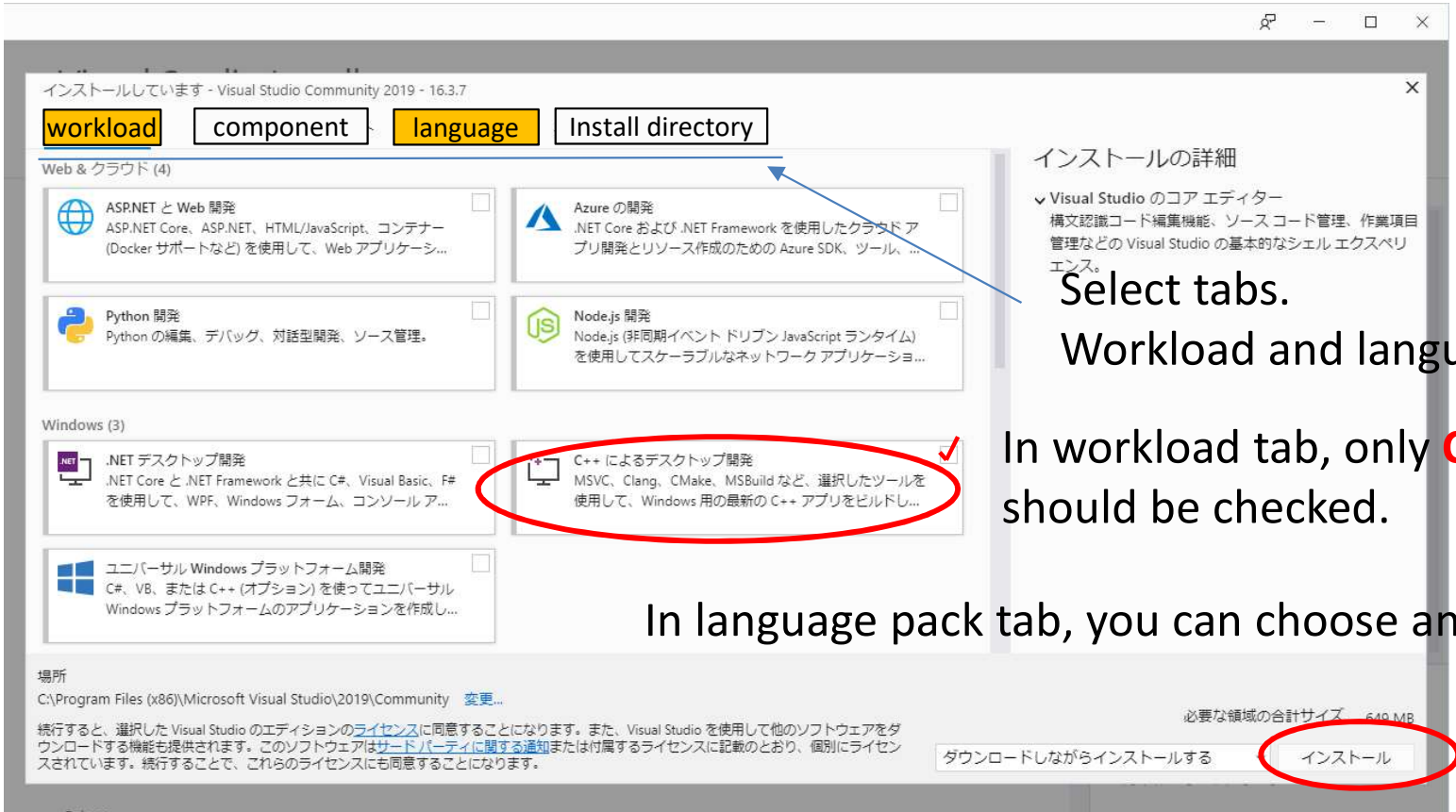
How to install Visual Studio Community

'Visual Studio **community** 2019' can select programing language or development hardware such as web application, android or iOS.

You can learn Visual Studio Community below
https://docs.microsoft.com/en-us/visualstudio/?view=vs-2019

Download '**vs_Community.exe**' which is an on-line install application of the up-to-date version of visual studio community (Now → 2019)

Execute 'vs_Community.exe'.　After system check, the following window will open.



The language of the installer depends on language environment of your PC.

Select tabs.
Workload and language are important.

In workload tab, only **C++ desktop environment** should be checked.

In language pack tab, you can choose any multiple languages.

Then, install

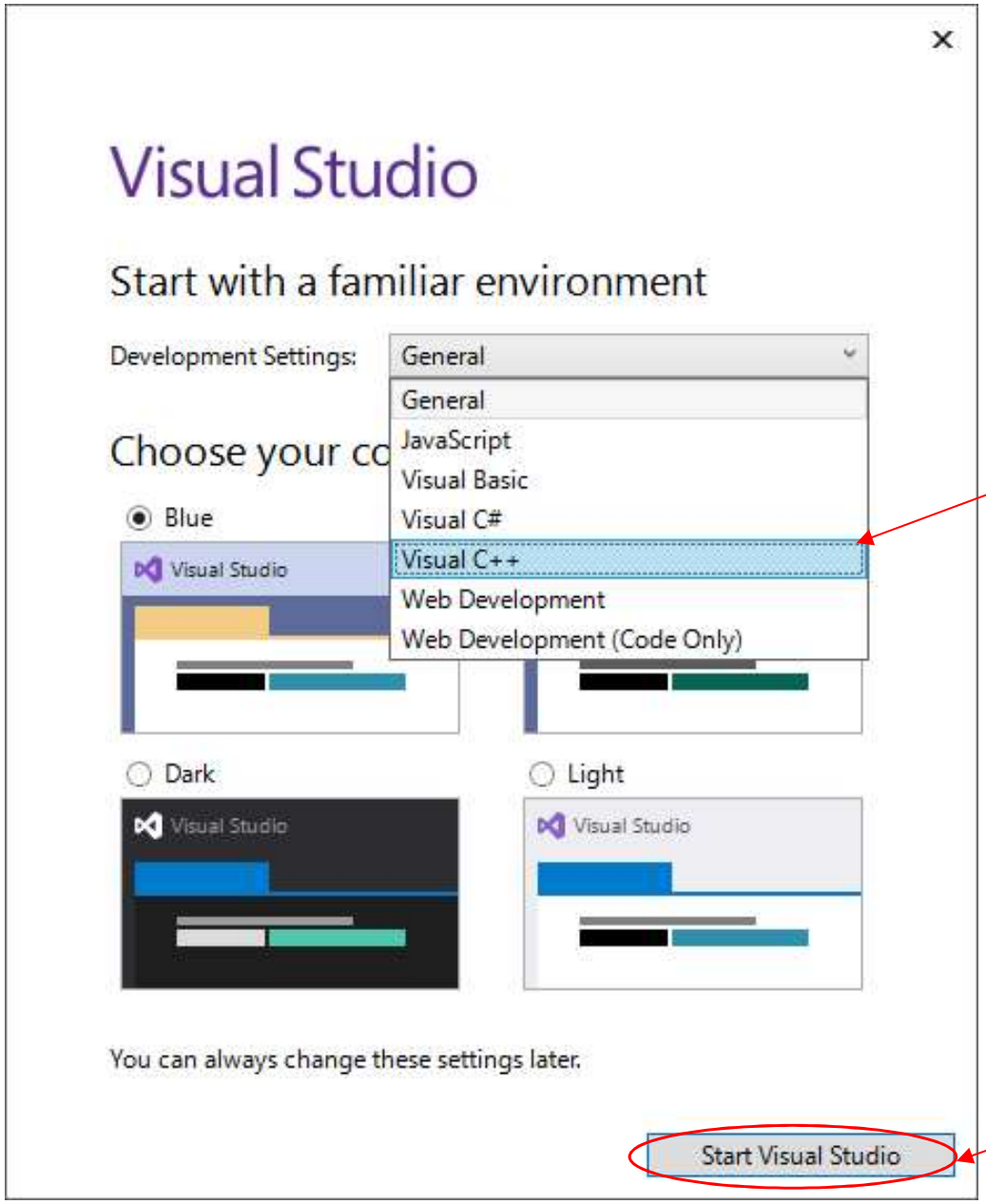
How to install Visual Studio Community

Wait for 30 minutes

After completing install and restarting PC, you can find the following application in start menu.

How to start Visual Studio Community 2019

Hereafter, the instructions is done under Visual Studio Community.



When you firstly start Visual Studio Community, passing sign-up dialog,

the following dialog appears.

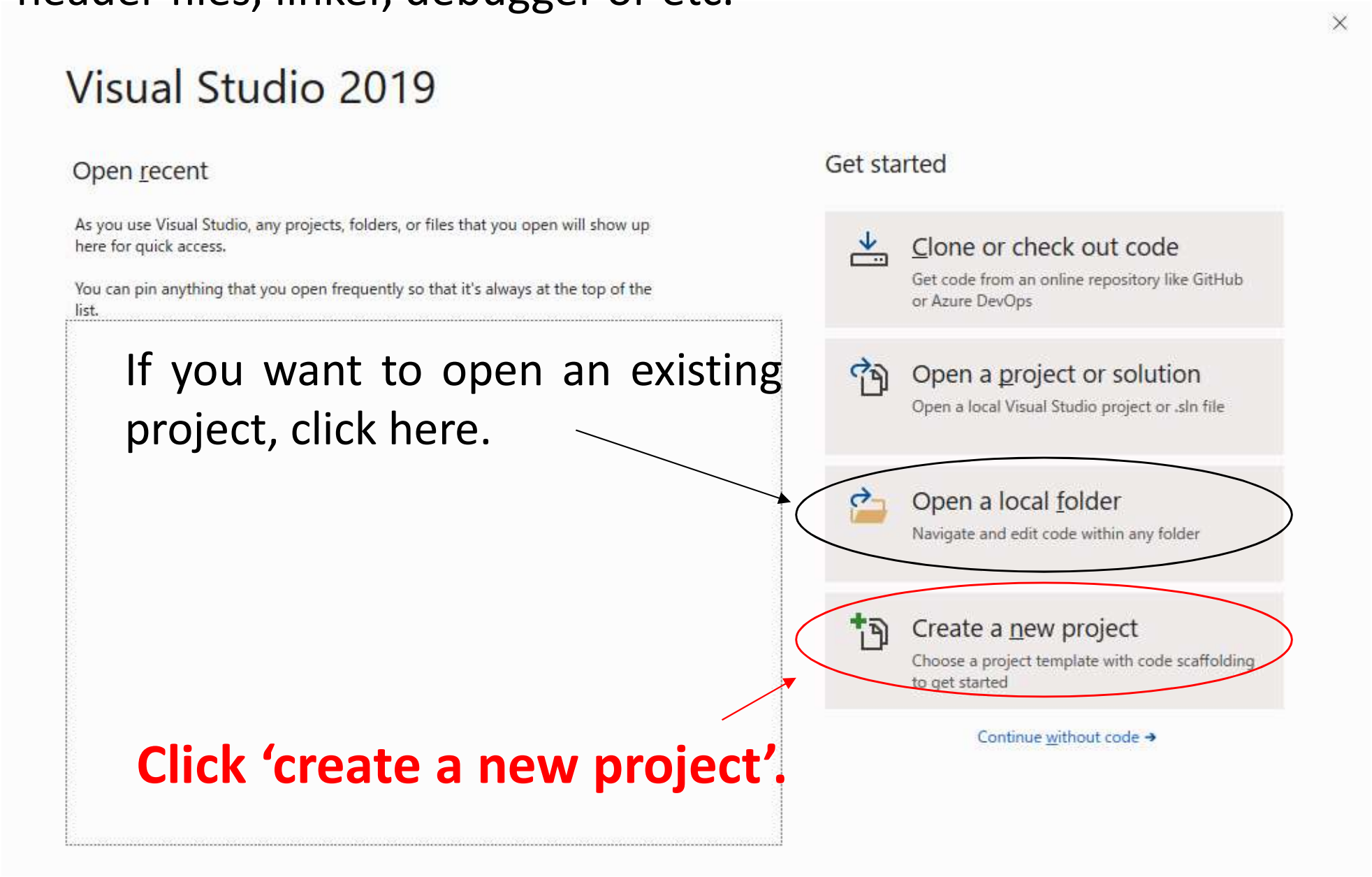
Choose Visual C++.

When you make an executable application working on Windows, a project for visual C++ should be created for one application.

Start Visual Studio.

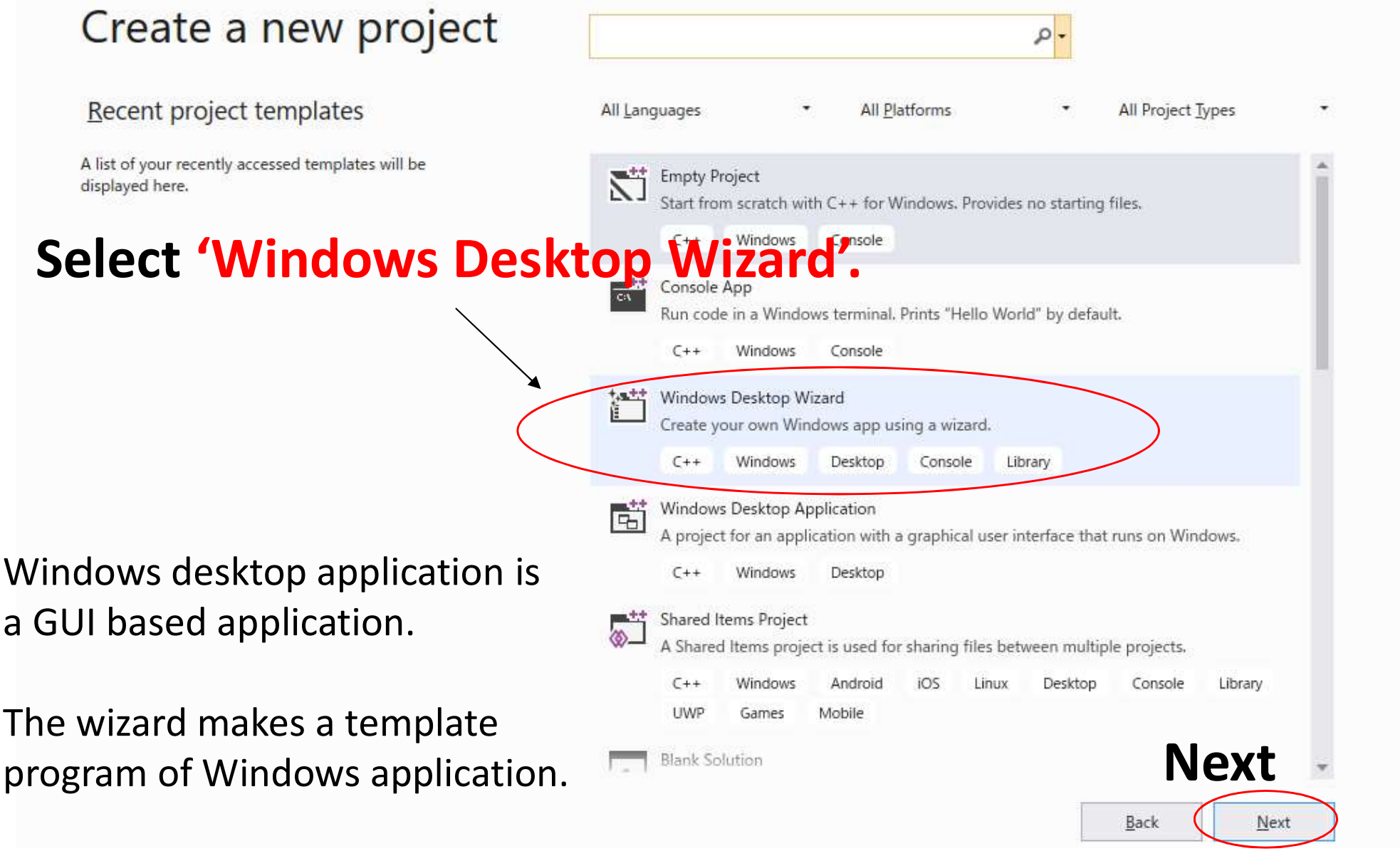
Project wizard

A project in ‘Visual Studio’ is container including C-source files, header files, linker, debugger or etc.



Creating new project(1)

Caution: you have to strictly follow the following instruction to create new project.



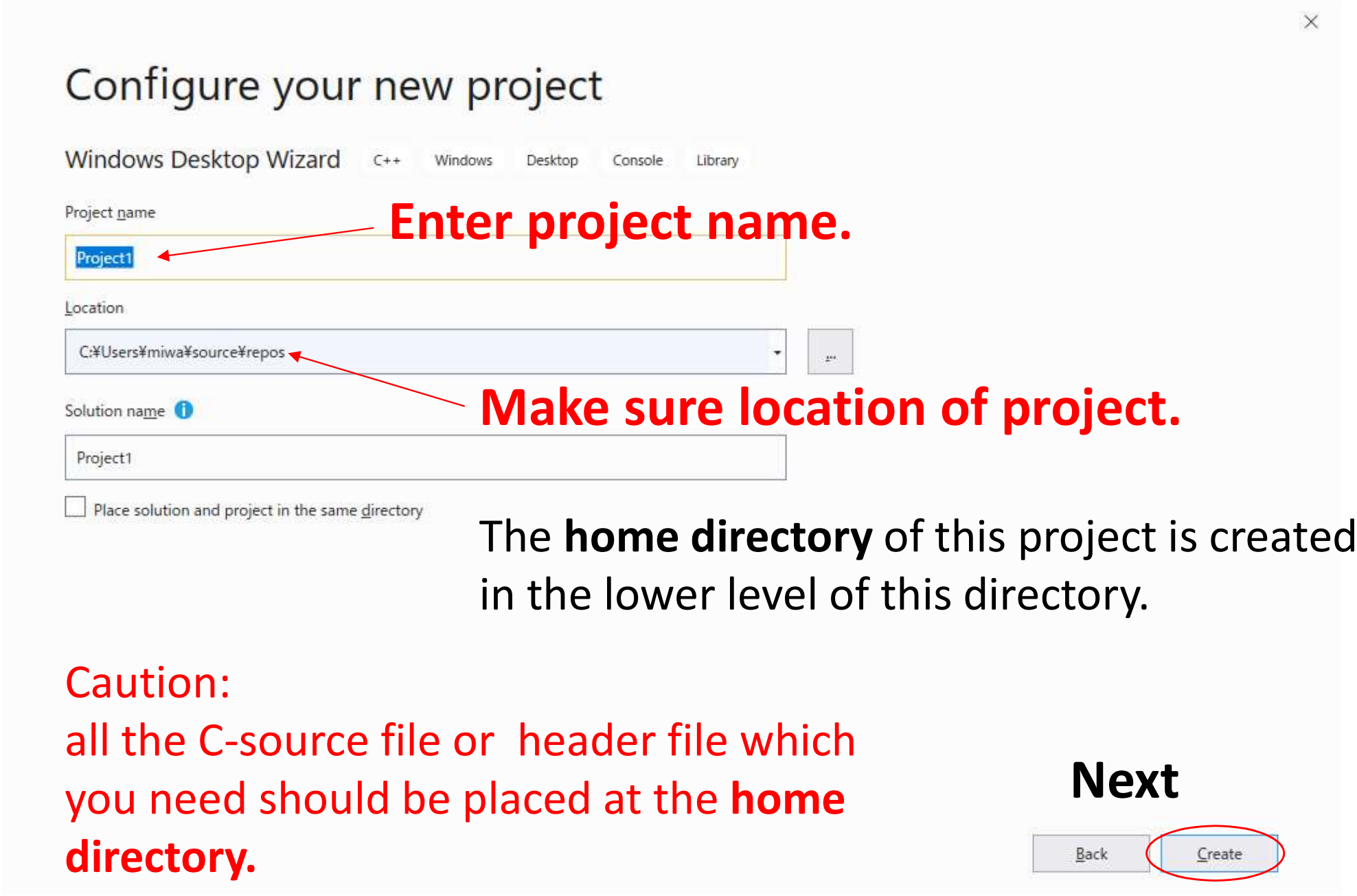
Select ‘Windows Desktop Wizard’.

Next

Windows desktop application is a GUI based application.

The wizard makes a template program of Windows application.

Creating new project(2)



The **home directory** of this project is created in the lower level of this directory.

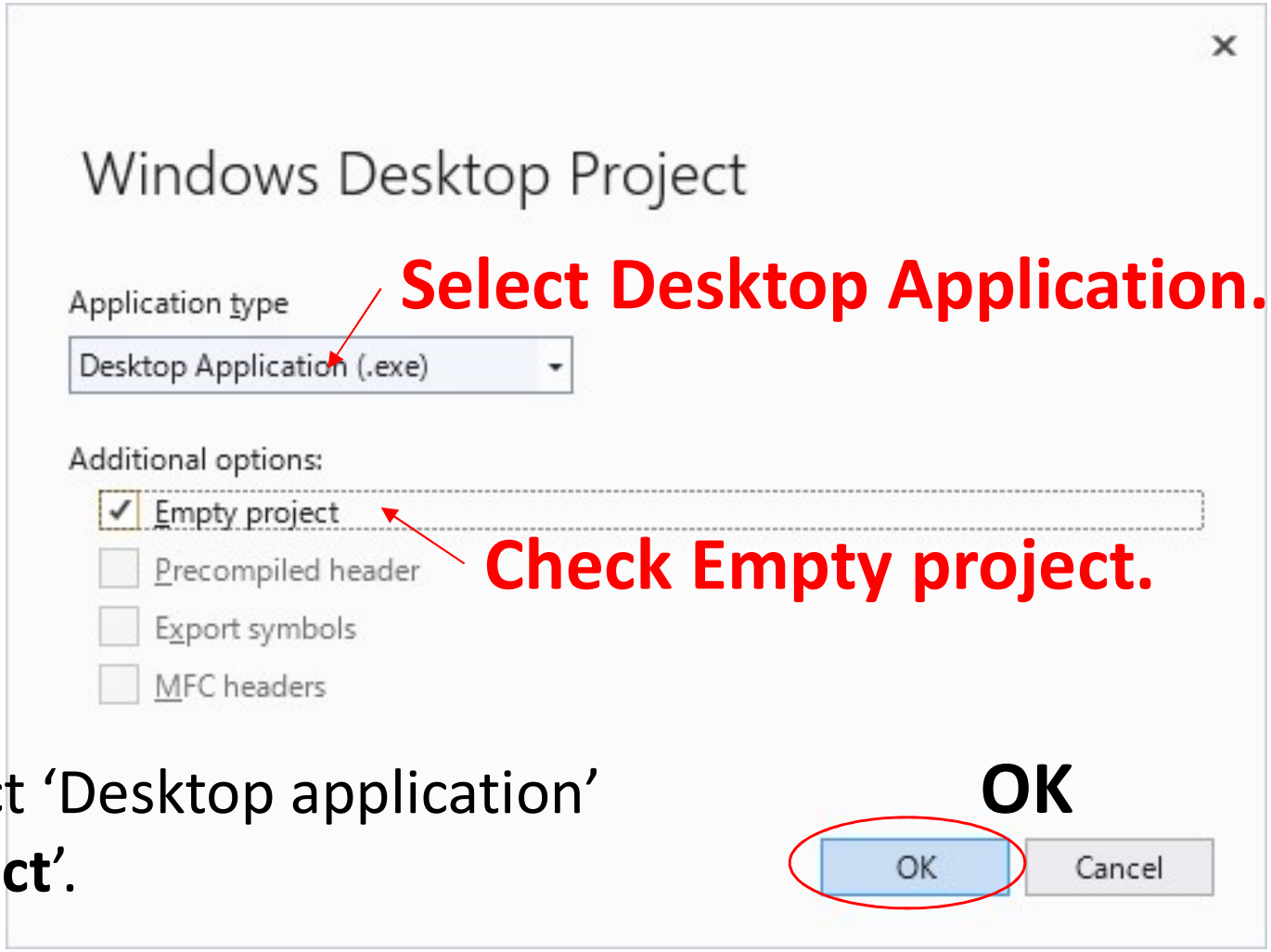
Caution:
all the C-source file or header file which you need should be placed at the **home directory.**

Creating new project(3)

Caution: you have to strictly follow the following instruction to create new project.

Mlib is C-based header files with many functions for visualization.

It also plays a role as a template program of **Desktop application**.



Here, you have to select ‘Desktop application’ and check ‘**Empty project**’.

How to install Visual Studio Express 2017

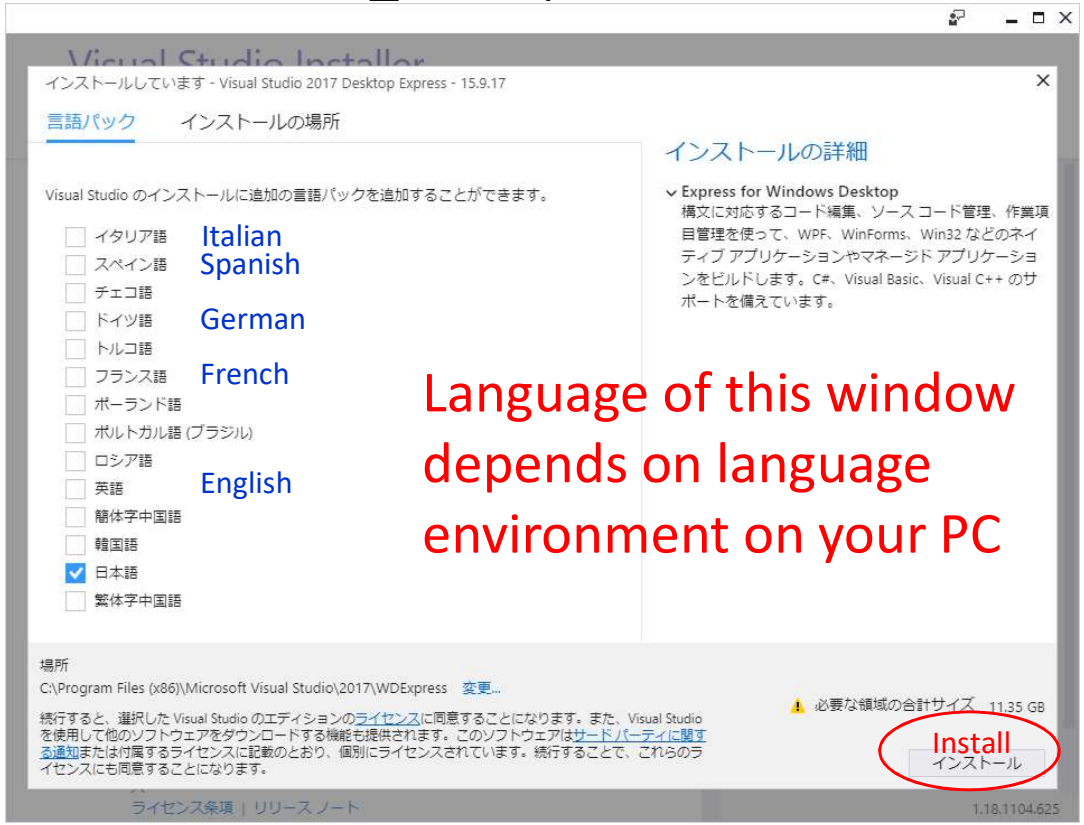
Visual Studio Community 2019 is up-to-date version. But ‘Visual Studio **Express** 2017’ is also available.

Here, it is introduced how to install ‘Visual Studio **Express** 2017’

You can easily find ‘Visual Studio Express 2017’on web search.

Executable file ‘vs_WDExpress.exe’ is an installer for multi languages.

After install vs_WDExpress.exe, the following window will open.

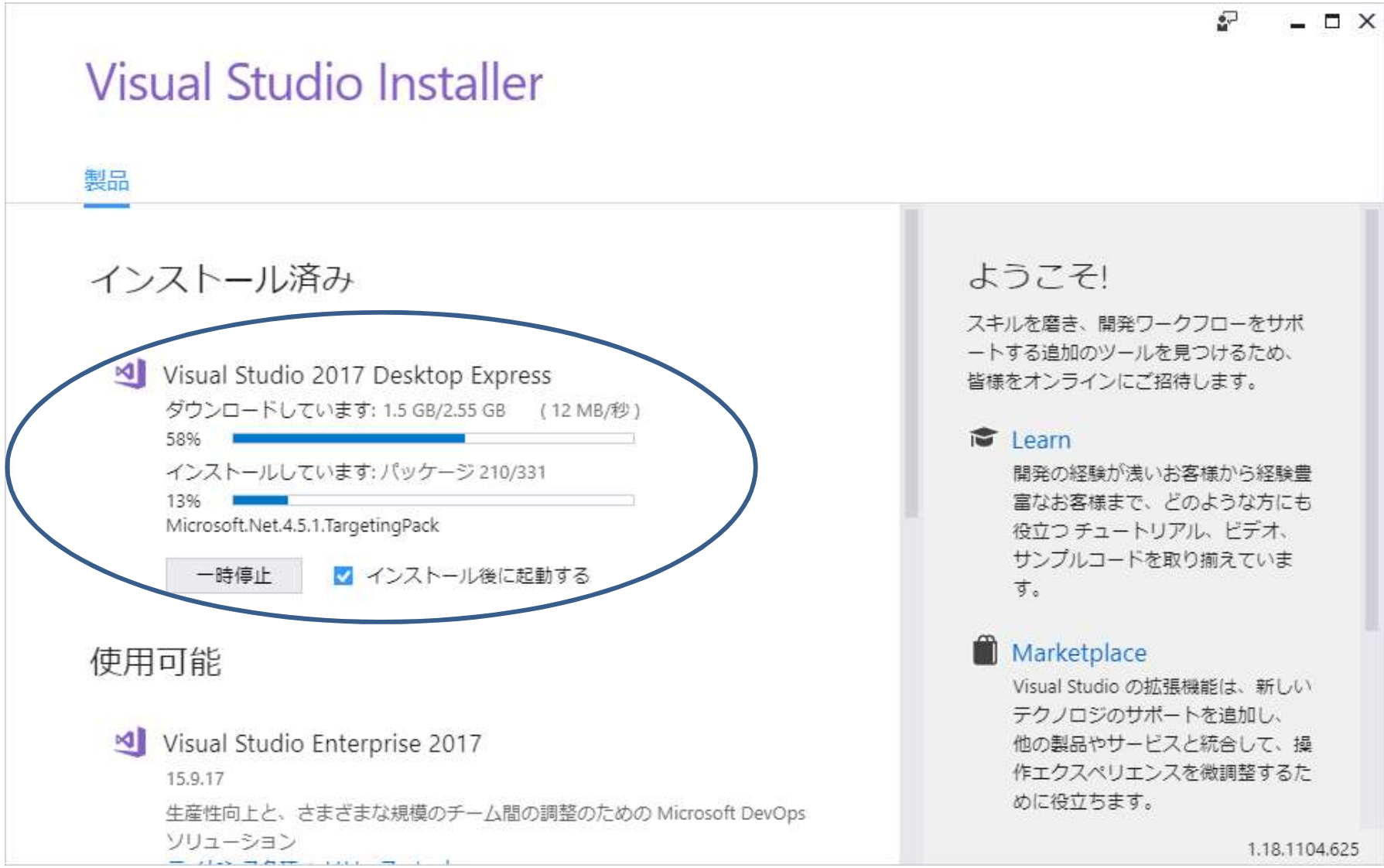


Please select your favorite language.

Available for multiple languages

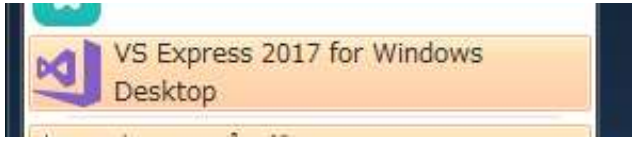
Click Install

How to install Visual Studio Express 2017



Wait for 20 or 30 minutes

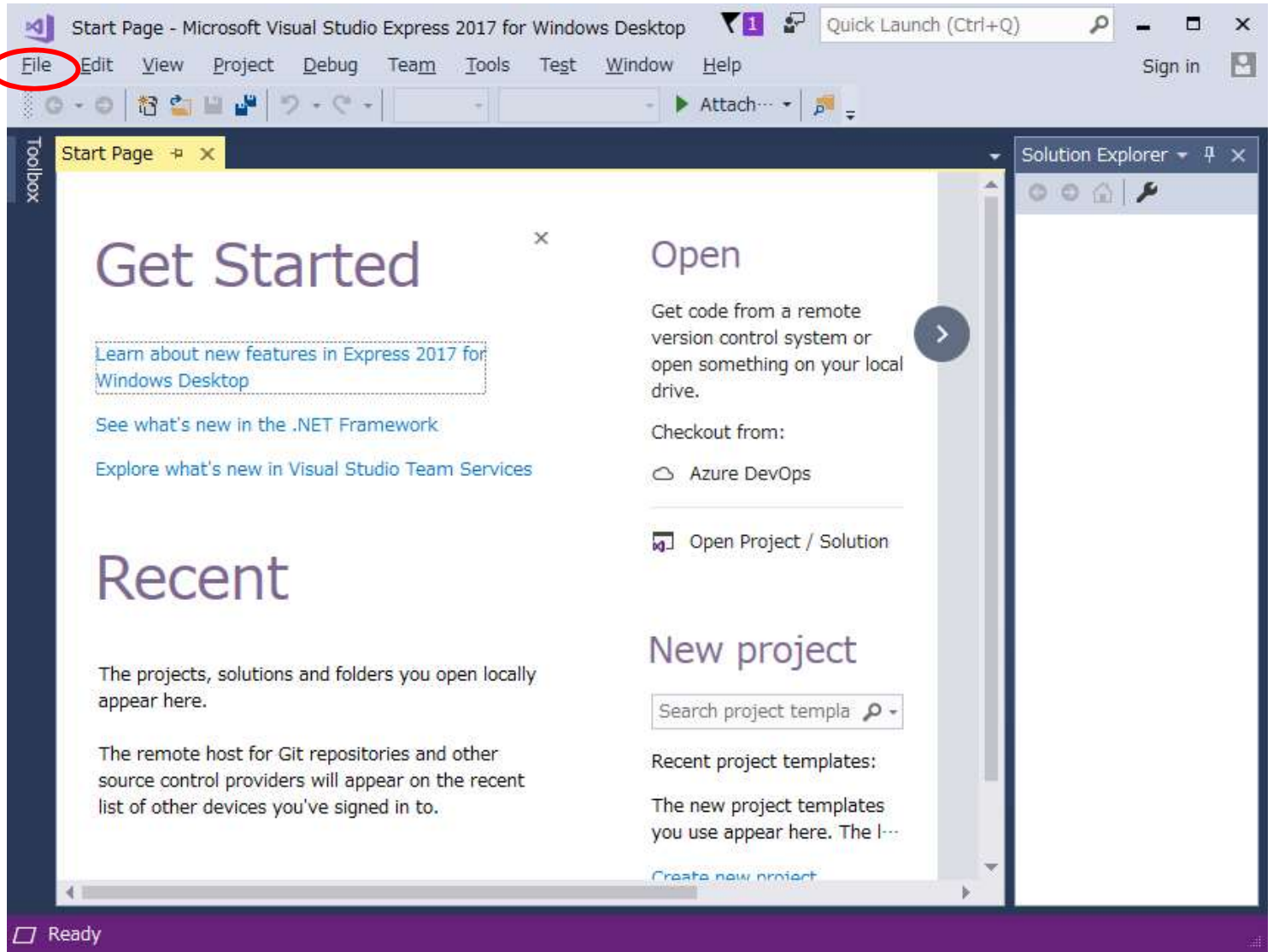
After completing install, you can find the following application in start menu



How to start Visual C++

Whichever you install, the following instruction is same.
When you make an executable application working on Windows,
One project is needed for one program.

Left click.
Select Create New Project.

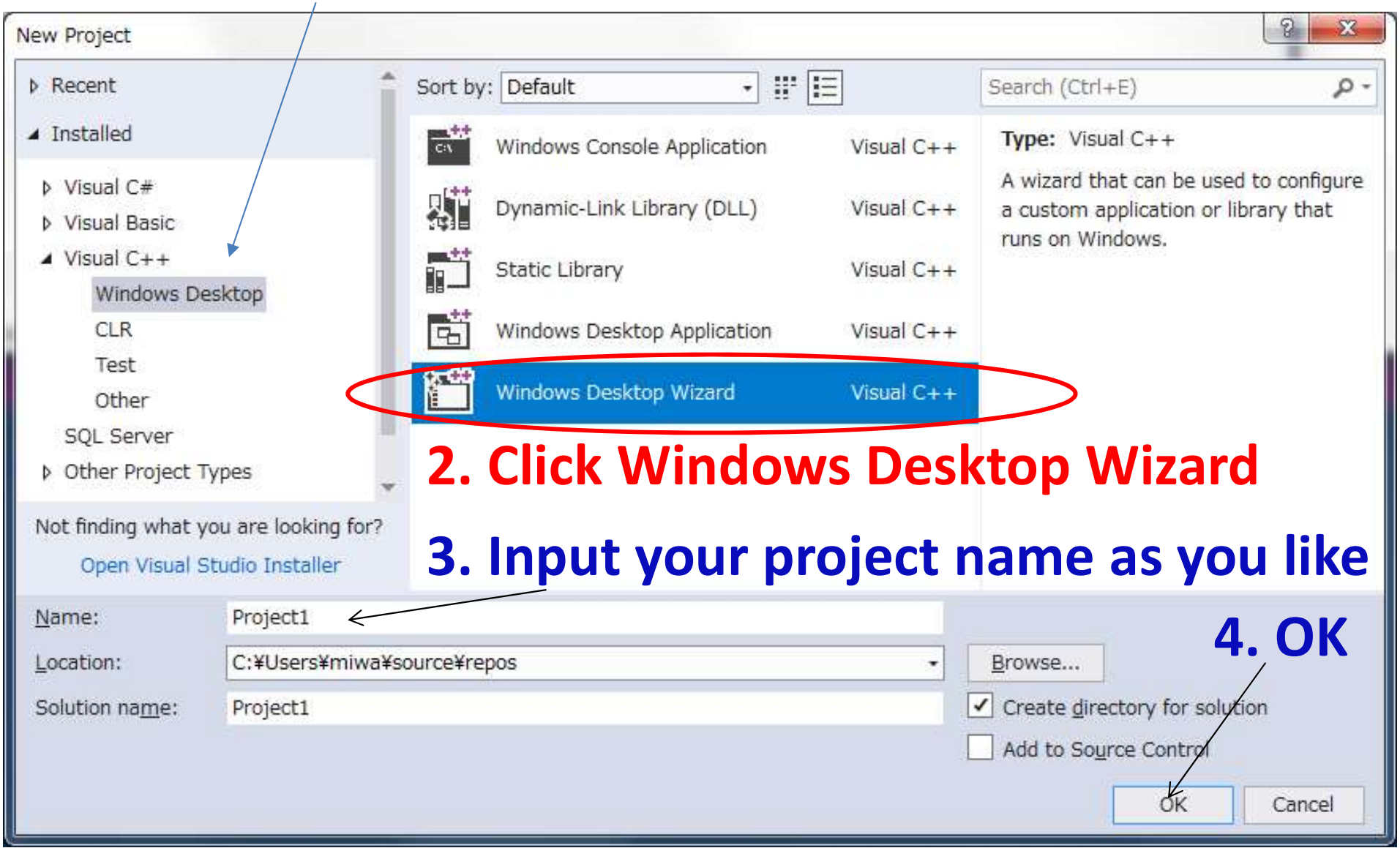


Project in Visual Studio is container including c-source files, header files, linker, debugger or etc.

Creating new project

Caution: you have to strictly follow the following instruction to create new project

1. Select Visual C++ → Click Windows Desktop

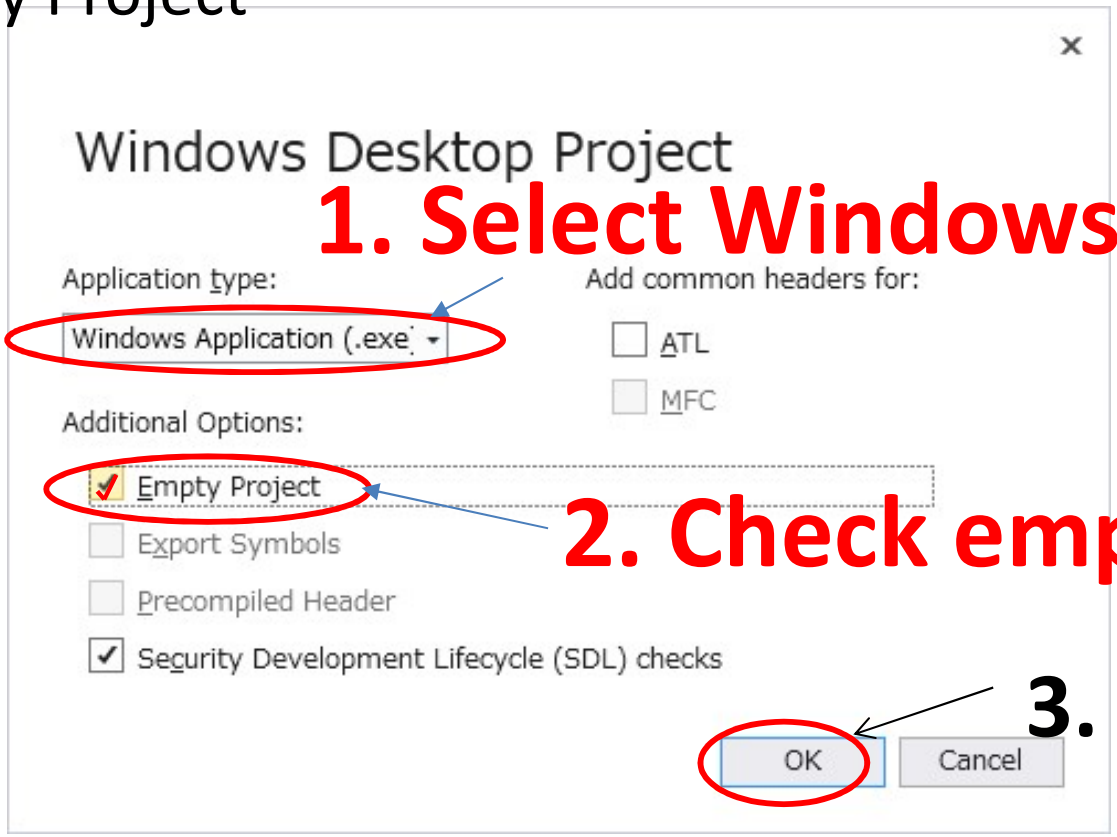


Windows desktop wizard

Windows desktop application is a GUI based application.
The wizard makes a template program of Windows application.

Mlib is C-based header files with many functions for visualization.
Moreover, it also plays a role as a template of Windows application.

Here , you have to select ‘Windows application’ and
check ‘Empty Project’

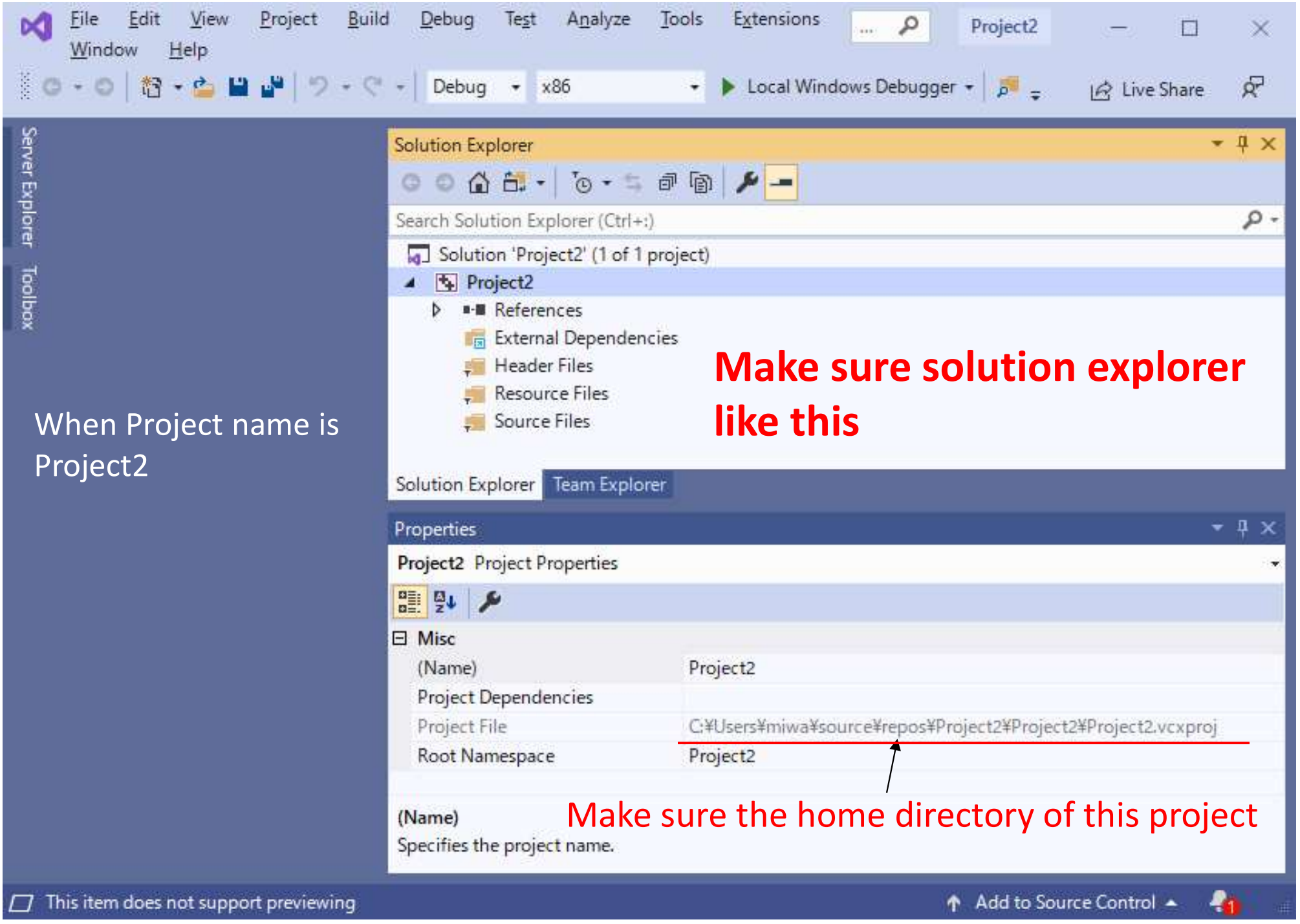


1. Select Windows application

2. Check empty project

3. Complete

Initial screen of project



When Project name is Project2

Make sure solution explorer like this

Make sure the home directory of this project

Confirmation of project creation

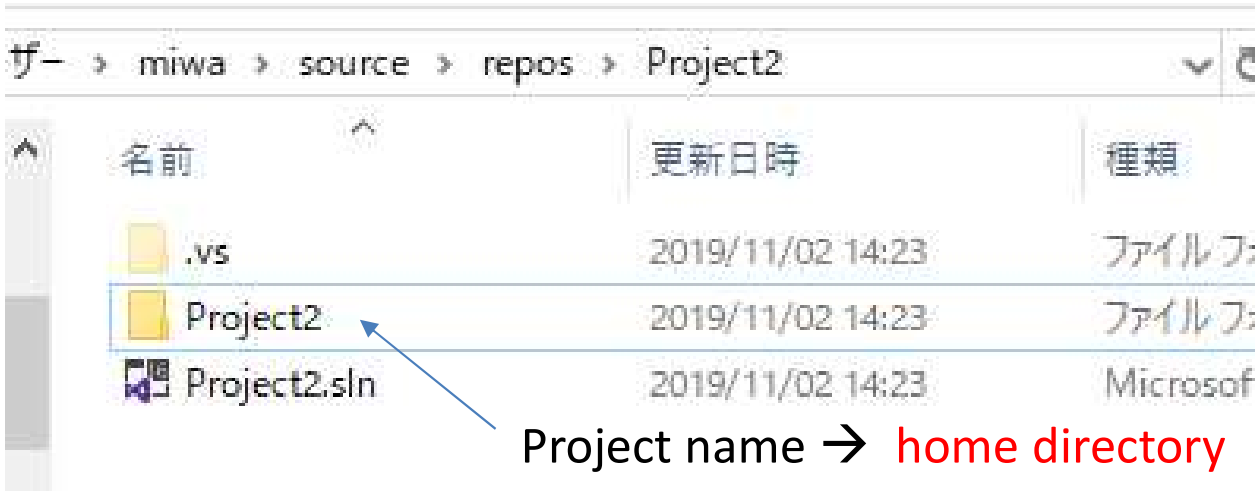
You have to confirm the location of the home directory

C:¥Users¥username¥source¥repos¥Project2¥Project2

Caution: Not one or two level lower

Using Windows explorer, find your home directory.

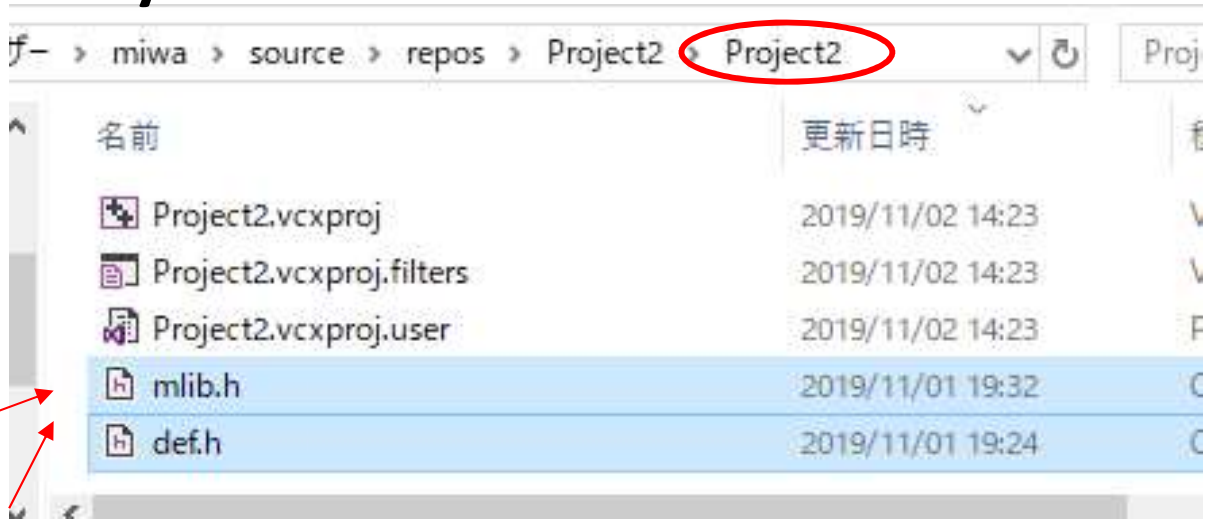
The project name is assumed to be ‘Project2’.



Project name → home directory

View inside the home directory.

Caution:
All the C-source file or header file which you need should be placed at the home directory.



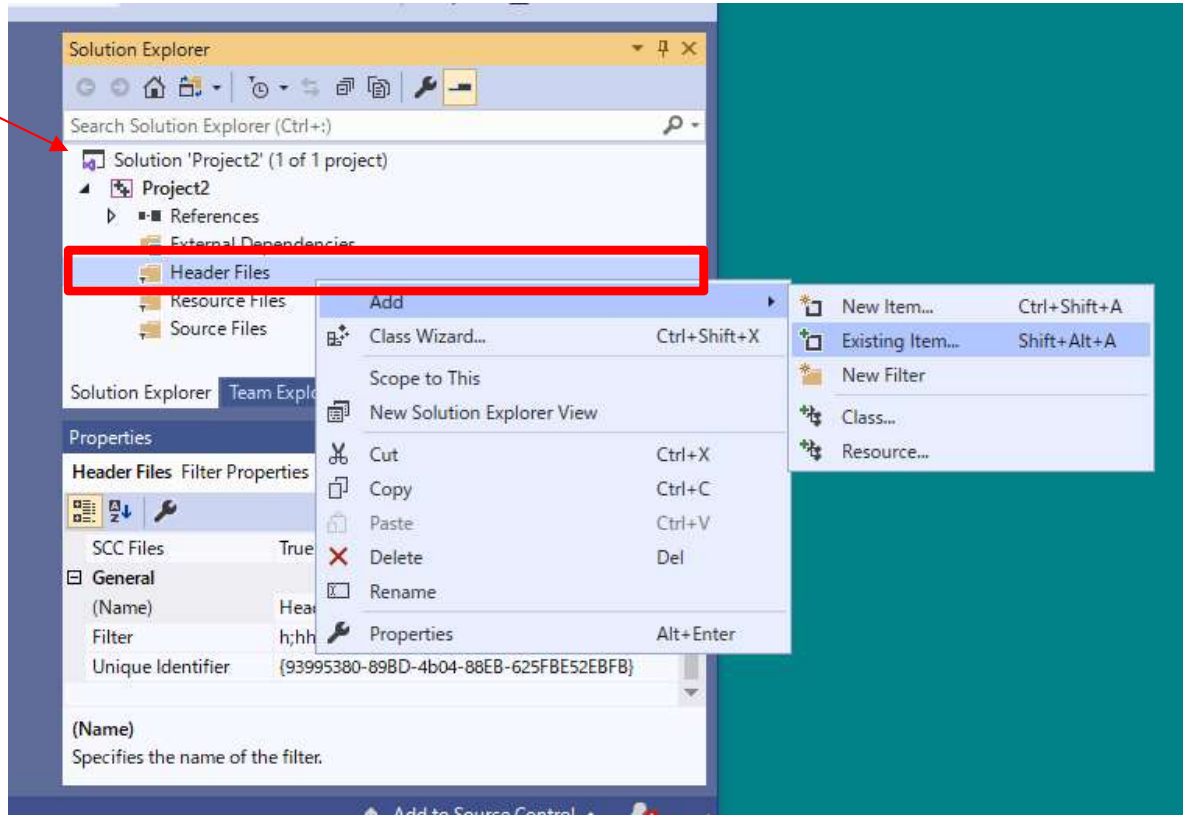
Add ‘mlib.h’ and ‘def.h’ here manually.

Registering files to project

As the project is empty, so some files should be registered in project.

Solution Explorer can view header files, resource files, source files to use for the project.

Register mlib.h and def.h in Header Files.



How to add

Right Click **Header files**

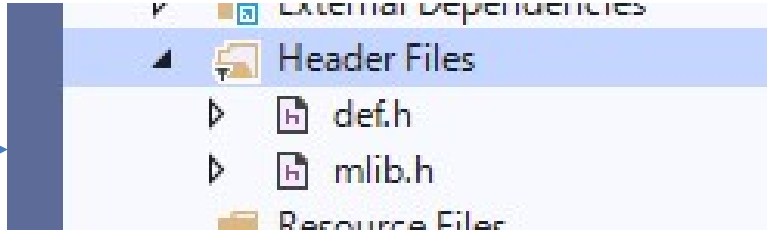
Select Add

Select Existing item

Select mlib.h and def.h copied at home directory

If you use other header files, you should also copy them at the home directory and add them in **Header Files.**

After registration, like this.



Registering new files to project

Create an original C-source file.

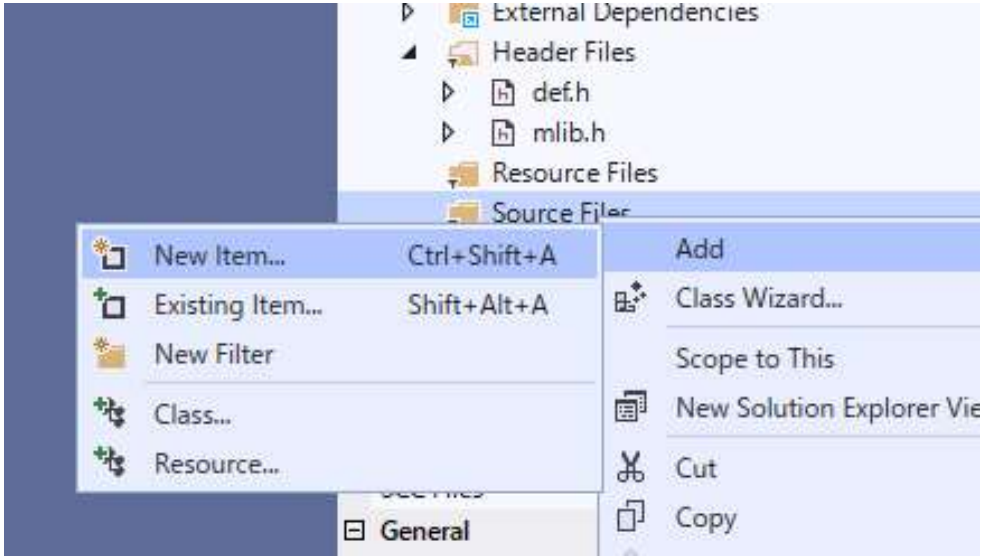
How to add

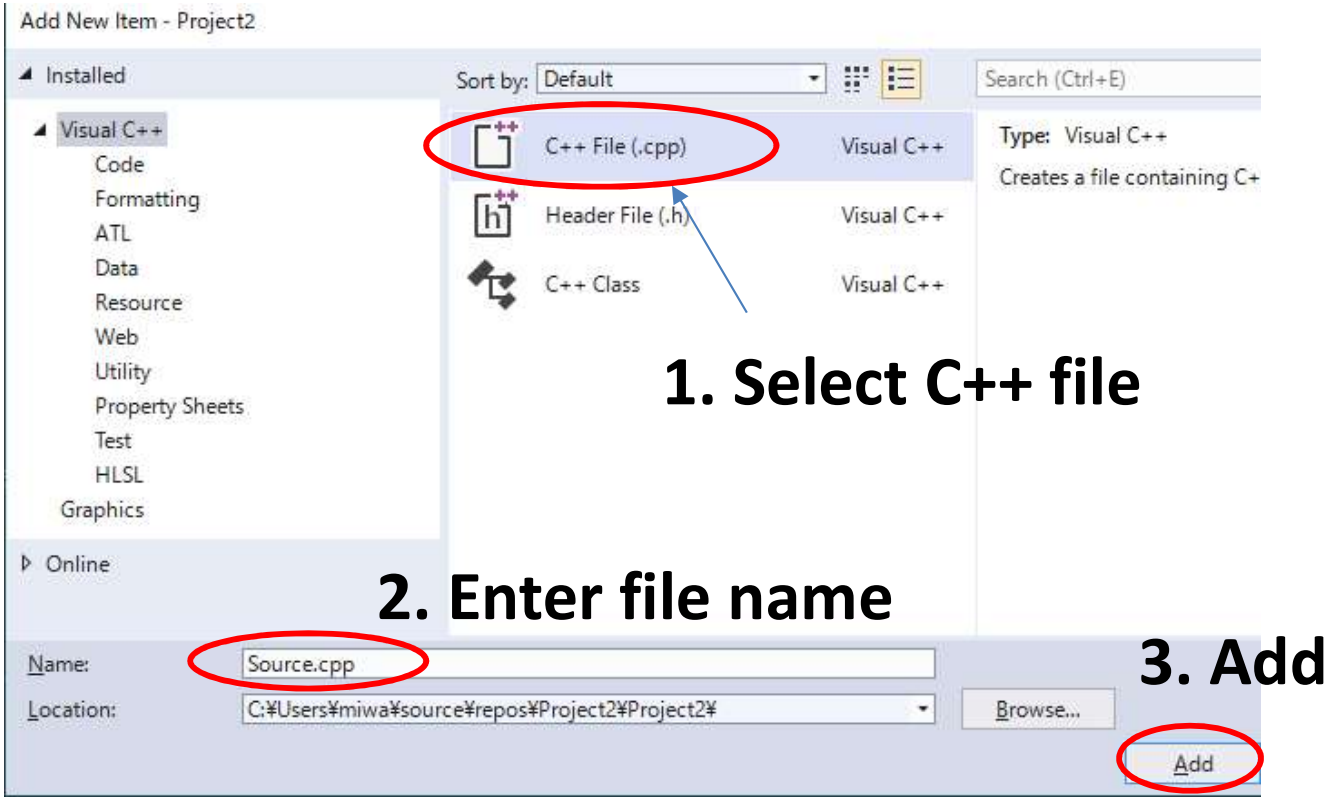
Right Click **Source files**

Select Add

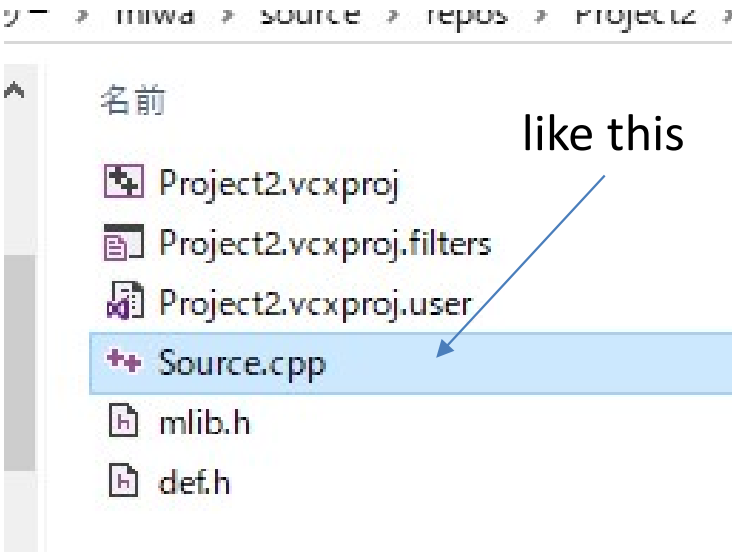
Select New item

'Add new item wizard' open



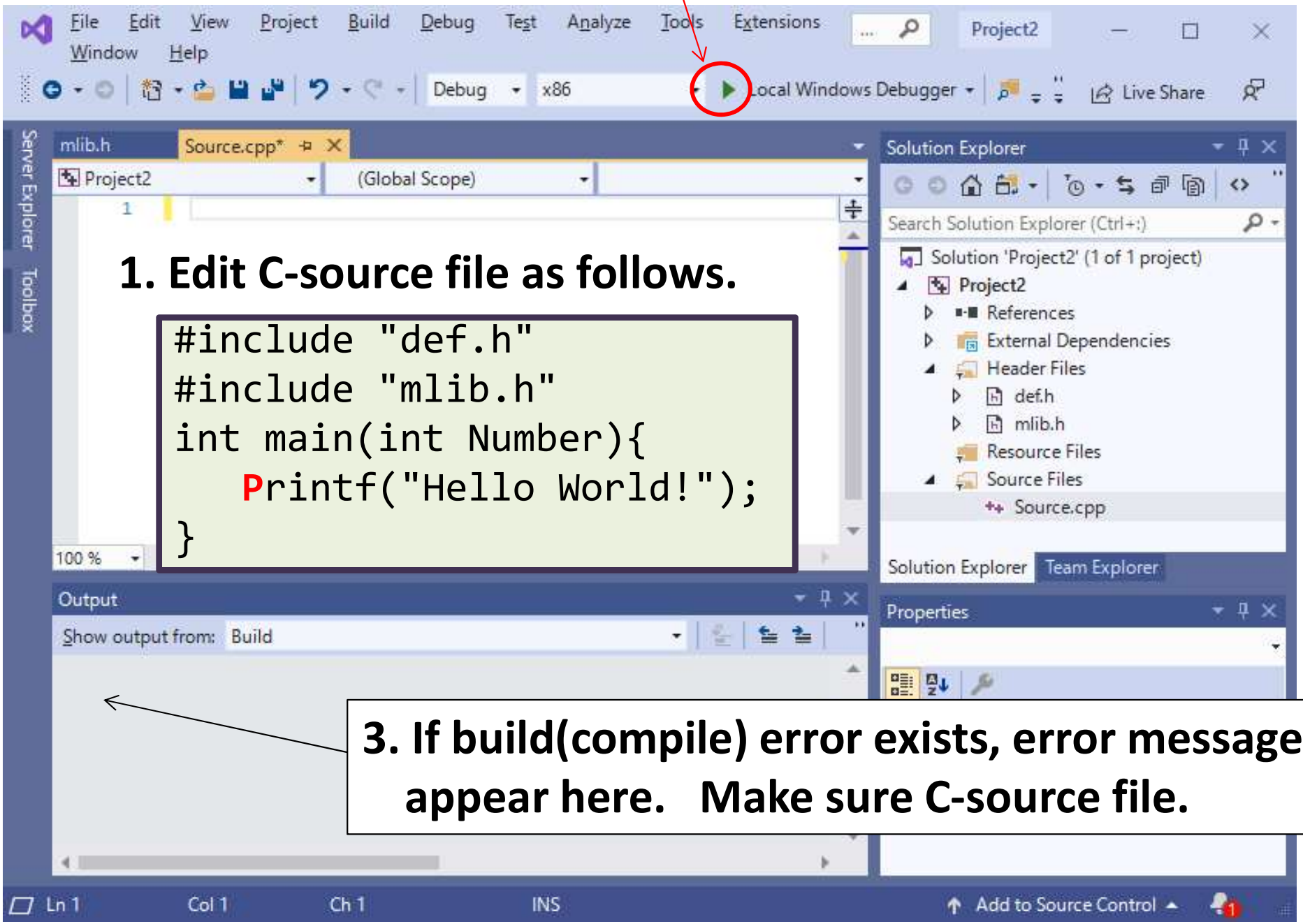


A new source files is created in the home directory.



Editing program and build(compile)

2. Start compiling and executing by this button.

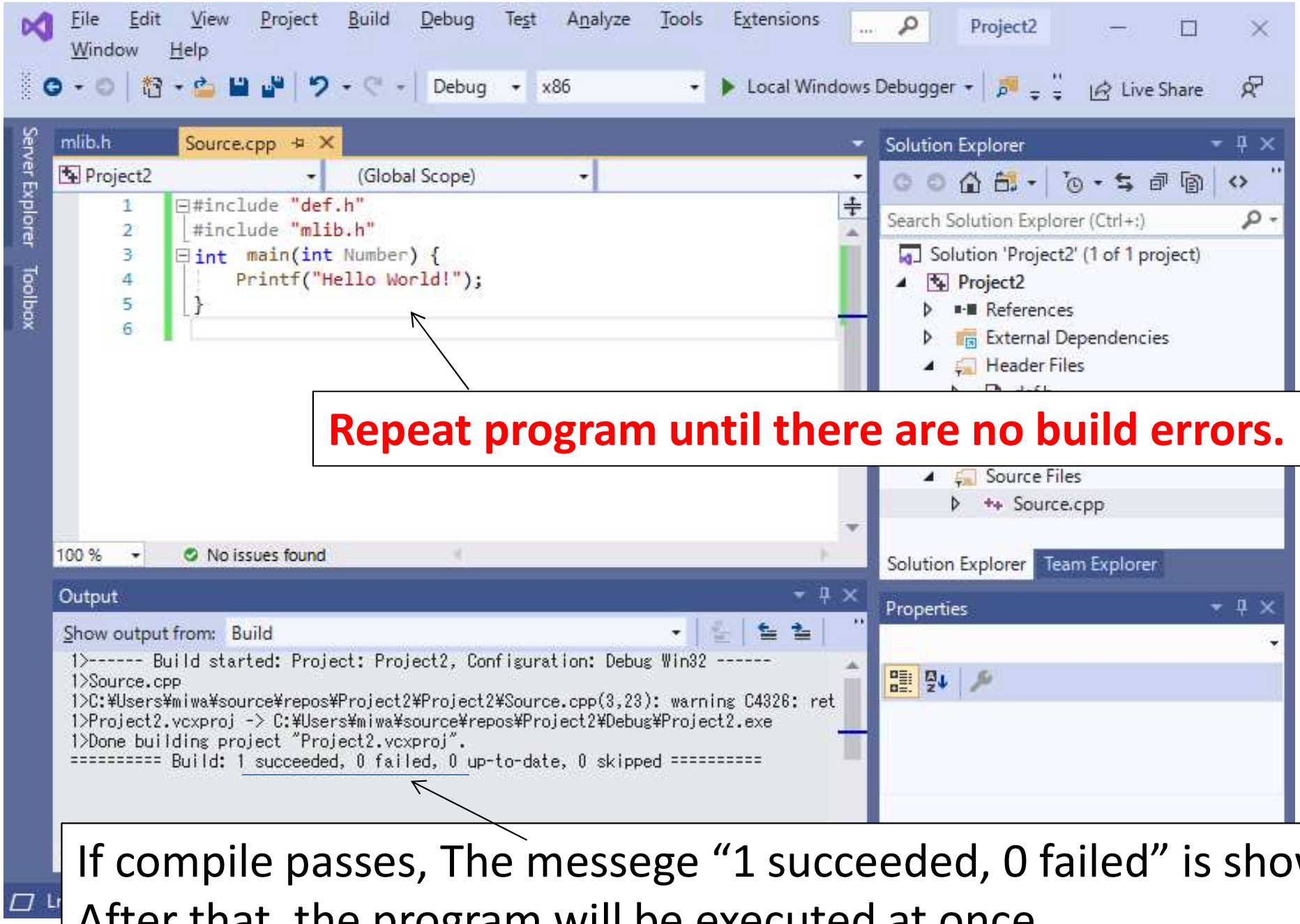


1. Edit C-source file as follows.

```
#include "def.h"
#include "mlib.h"
int main(int Number){
   Printf("Hello World!");
}
```

3. If build(compile) error exists, error messages appear here. Make sure C-source file.

Building(Compiling)

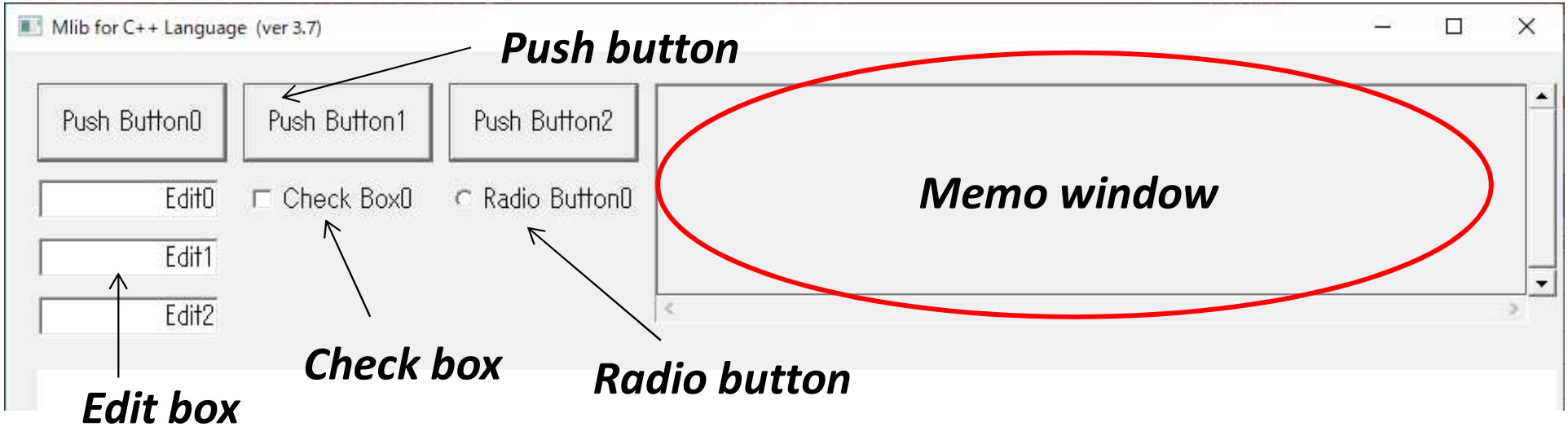


Repeat program until there are no build errors.

If compile passes, The messege “1 succeeded, 0 failed” is shown.
After that, the program will be executed at once.

Executing Mlib

The following window opens. This is a basic window of MLIB.



When you click any pushbutton, the program is executed.

“Hello World!” is displayed in the memo window.

How to use Mlib

def.h manages constant, global variables and window conditions such as the number, position and size of the pushbutton, edit box, check box etc.

mlib.h receives and processes messages for window action from Windows system and includes many function groups for data visualization and drawing a simple shape and text.

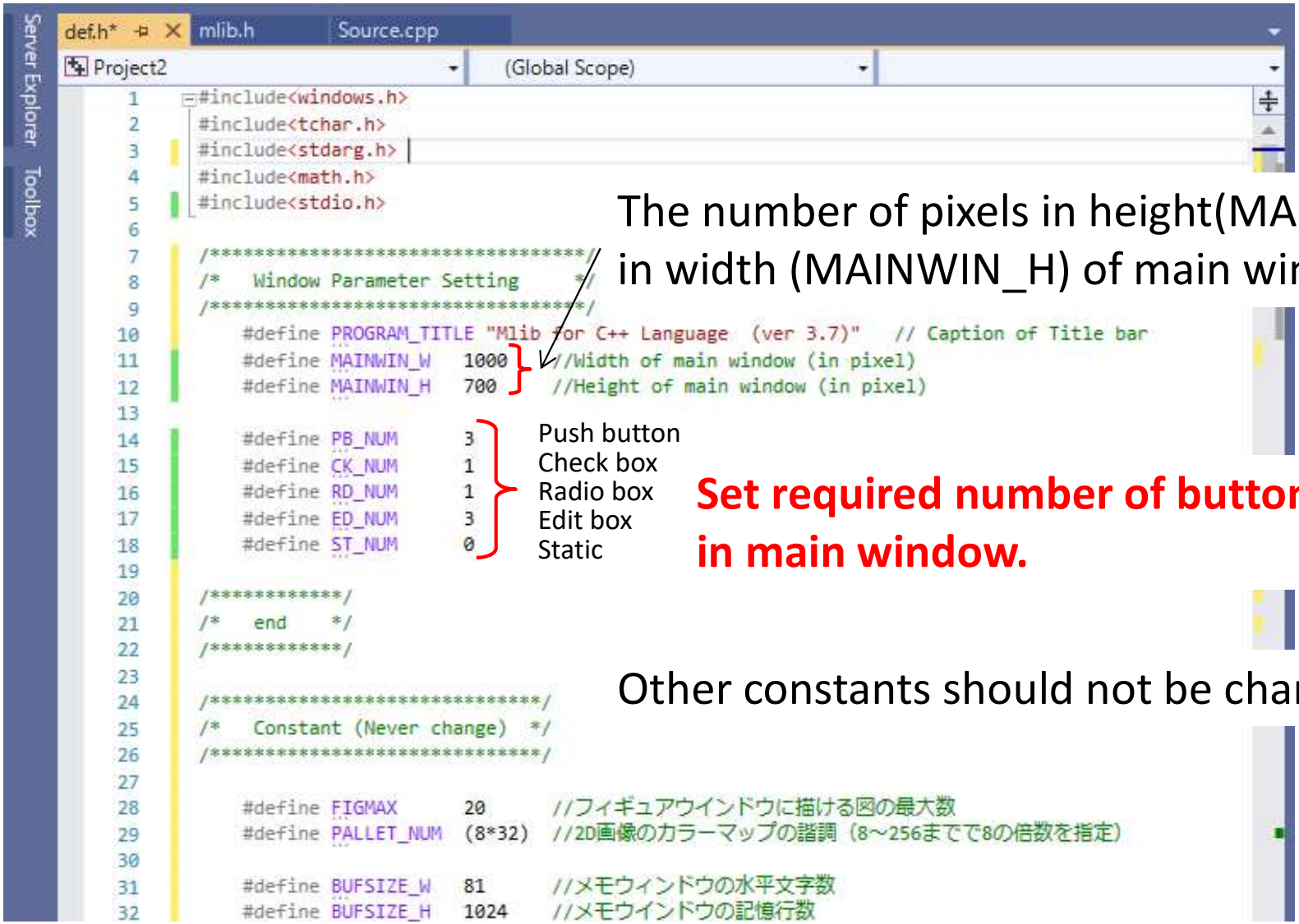
source.cpp is a main program which you should edit.
The main program should include one ‘main()’ function.
When any pushbutton is pressed, the ‘main()’ function is executed.

The numerical value (0,1,2..) for pushbutton number is passed to argument ‘Number’ of ‘main()’ function.
The processing in each pushbutton can be changed by the numerical value of argument ‘Number’ .

Caution: In source.cpp, the include statement for def.h should be ahead of that of mlib.h.

Initial setting of window

def.h sets an initial condition of the window.



The number of pixels in height(MAINWIN_W) and in width (MAINWIN_H) of main window.

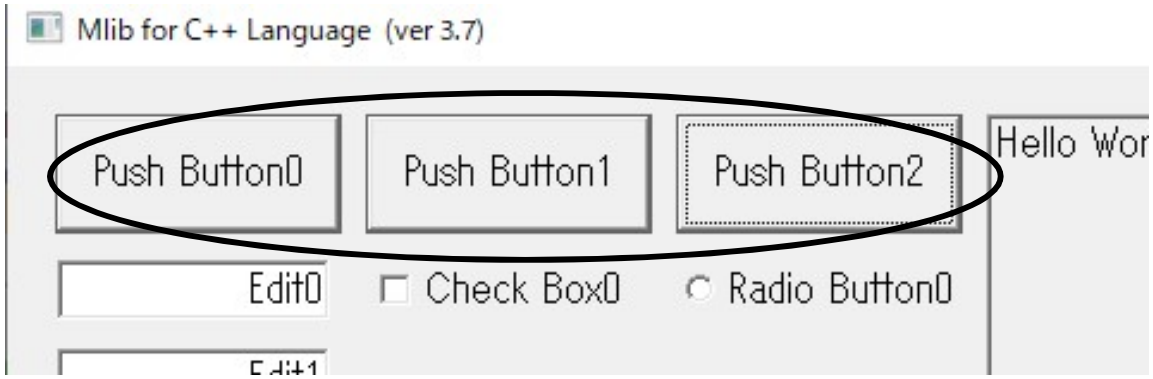
Push button
Check box
Radio box
Edit box
Static

Set required number of buttons or boxes in main window.

Other constants should not be changed.

The position or size of each button can be changed by Components() function defined latter.

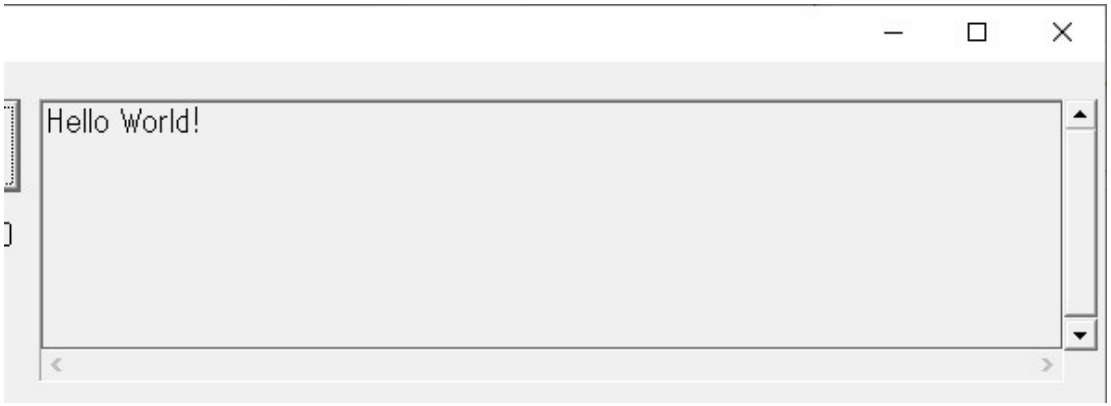
Push button



push button executes the main program.

If push button 1 is clicked, the argument ‘Number’ of ‘main’ function is set to be 1.

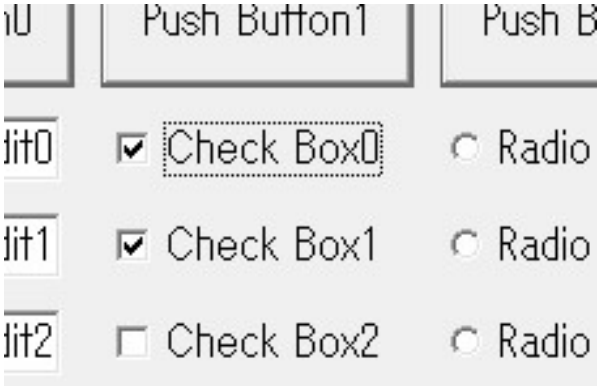
Memo window



Memo window is output area of **P**rintf() function which is compatible to printf() function.

This area can scroll down.
You can check the previous output by a scroll bar.

Check box



If you click the check box, a check mark is switched.

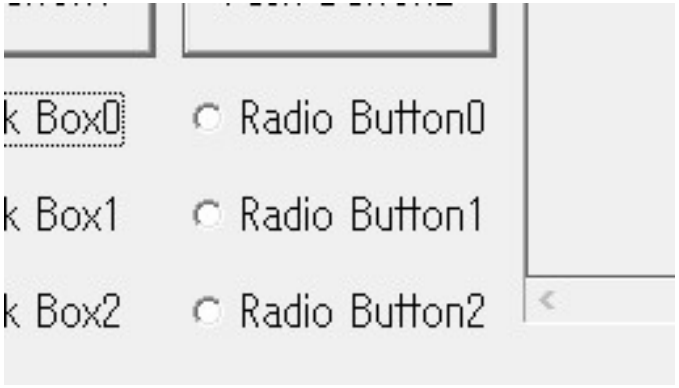
You can refer the check condition in main program.
by the global variables .

```
sCK[ checkbox number ].chk
```

When a check box is on or off, the value is 1 or 0, respectively.

In the above example, `sCK[0].chk=1` `sCK[1].chk=1` `sCK[2].chk=0`

Radio box



Some radio boxes should be grouping.
One is selected in the group by clicking the radio box .

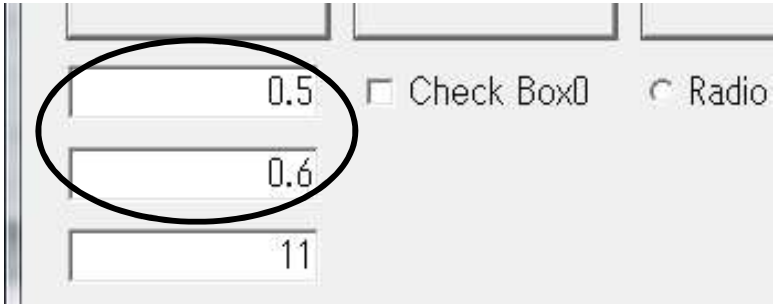
If you click a radio box, only the selected box is marked.

You can refer the select condition in main program.

```
sRD[ radiobox number ].chk
```

When a radio box is on and off, the value is 1 and 0, respectively.

EDIT BOX



Edit box will hand over a value entered in the box to main program.

It can repeatedly execute the main program with changing a parameter of the program.

```
int Get_int(int i)
```

```
b=Get_int(1);
```

```
double Get_double(int i)
```

```
a=Get_double(2);
```

These functions allow us to convert the string of the edit box specified by the argument ‘i’ to an integer or double precision value which is returned as a return value. When the string is not a numerical value, zero is returned.

```
void Set_double(int i , double x)
```

```
Set_double(0,1.2);
```

This function can output a double precision value to the edit box specified by the argument ‘i’ When the string is not a numerical value, zero is returned.

```
void Set_char(int i, char *Buffer)
```

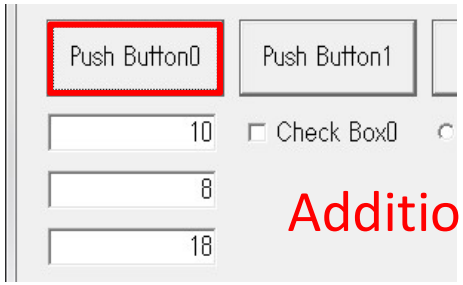
```
Set_char(0,”ABC”);
```

This function can output a string specified by character type array variable ‘Buffer’ to the edit box specified by the argument ‘i’.

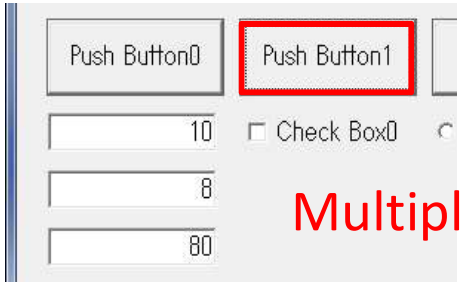
Example 1

Create a program that assigns the values entered in edit box #0 and edit box #1 to variables ‘a’ and ‘b’, respectively, and outputs the following calculation results in edit box #2.

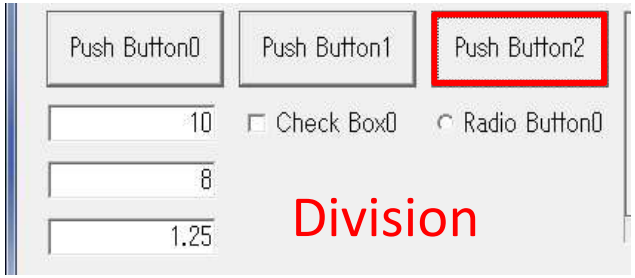
button #0 → a＋b



button #1 → a × b



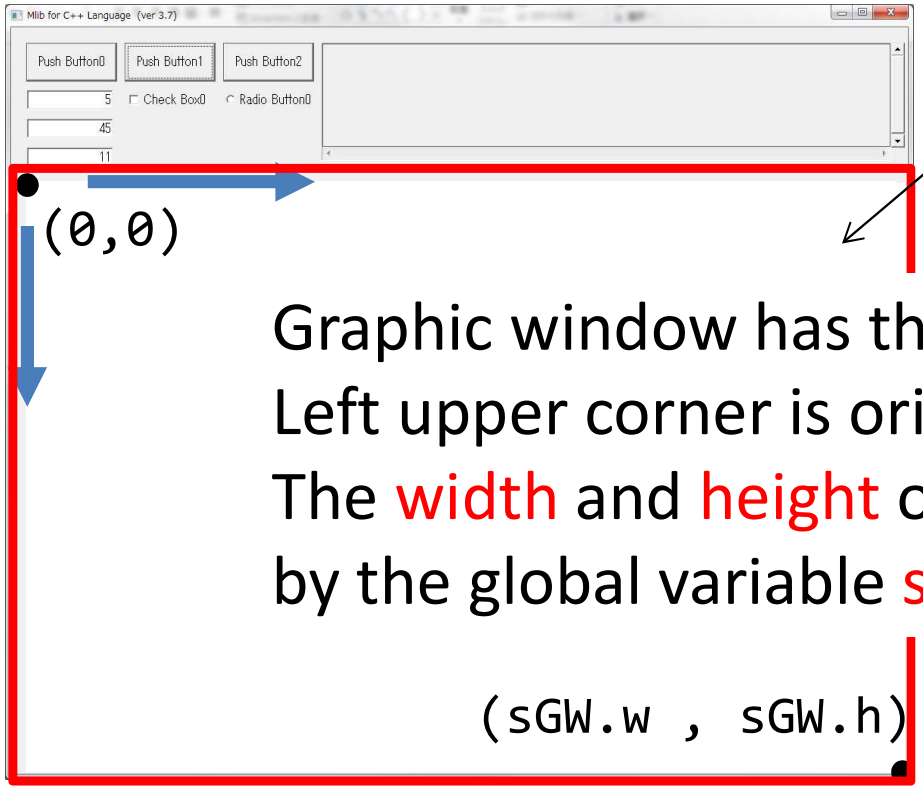
button #2 → a ÷ b




```
int main (int Number){
  double a,b;
  a=Get_double(0);
  b=Get_double(1);

  switch(Number){
    case 0:
      Set_double(2,a+b);
      break;
    case 1:
      Set_double(2,a*b);
      break;
    case 2:
      Set_double(2,a/b);
      break;
  }
}
```

Graphic Window



Graphic window

A window where the figure is drawn such as line, circle and rectangle.

Graphic window has the co-ordinate in pixel unit.
Left upper corner is origin point (0,0).
The width and height of graphic window can be referred by the global variable sGW.w and sGW.h, respectively.

Multiple figure windows can be created in the graphic window.

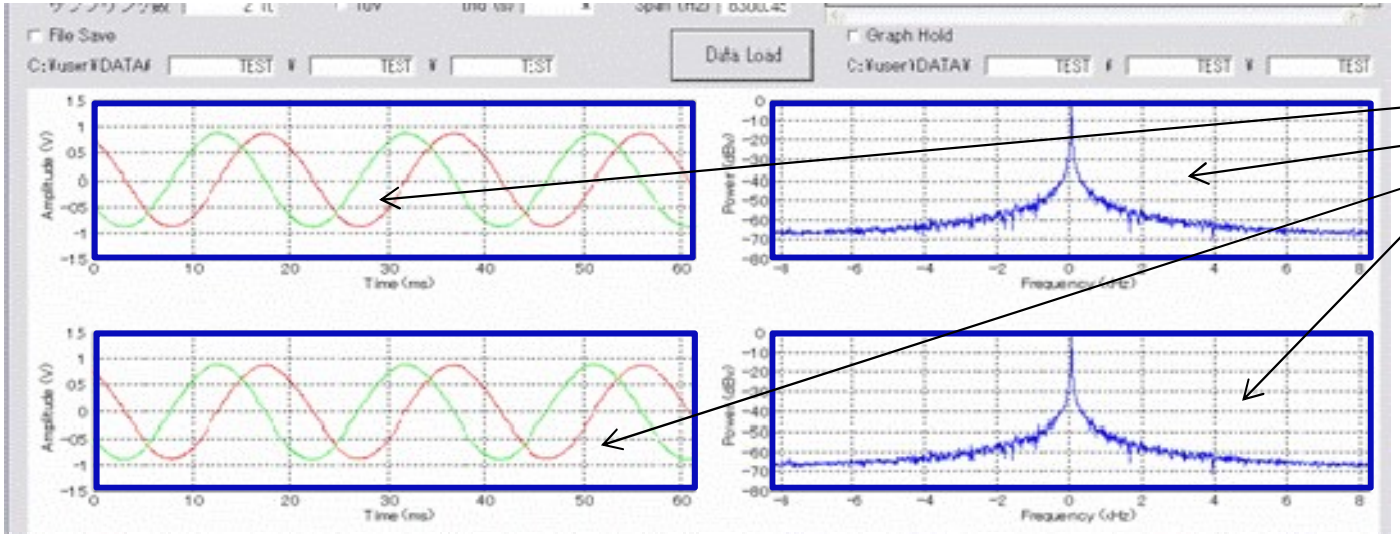


Figure window

Set_figure() function can create figure windows.

Each window has an individual x and y-axis.

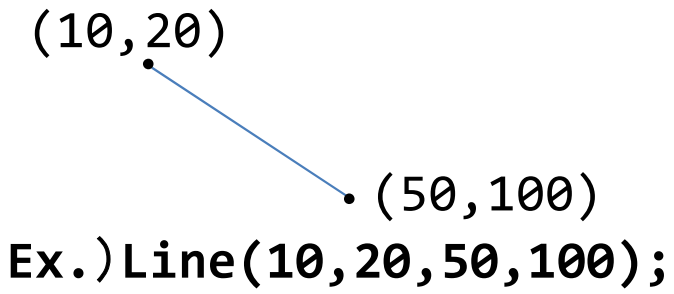
Free Drawing in Graphics Window

```
void Line(int x1,int y1,int x2,int y2)
```

Draw a straight line in the graphics window

(x1, y1) are the coordinates of the start point.

(x2, y2) are the coordinates of the end point.



Ex.)Line(10,20,50,100);

```
void Rect(int x1,int y1,int x2,int y2,int bfflug)
```

Draw a rectangle in the graphics window

(x1, y1) are the coordinates of upper left corner.

(x2, y2) are the coordinates of lower right corner.

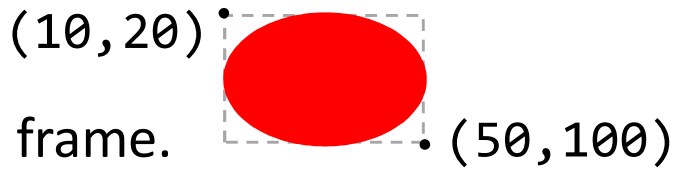
Ex.)Rect(10,20,50,100,1);


```
void Circle(int x1,int y1,int x2,int y2,int bfflug)
```

Draw an ellipse that touches the rectangle specified by coordinates in the graphics window.

Definition of coordinates is the same as above.

Ex.) Circle(10,20,50,100,1);



`bfflug`
- 0 : Only the frame. Do not draw inside the frame.
- 1 : Fill the frame with the specified color
- others: Fill the frame with the white color.

These function used colors and line type specified by Plot_pen() function

Creation of figure window

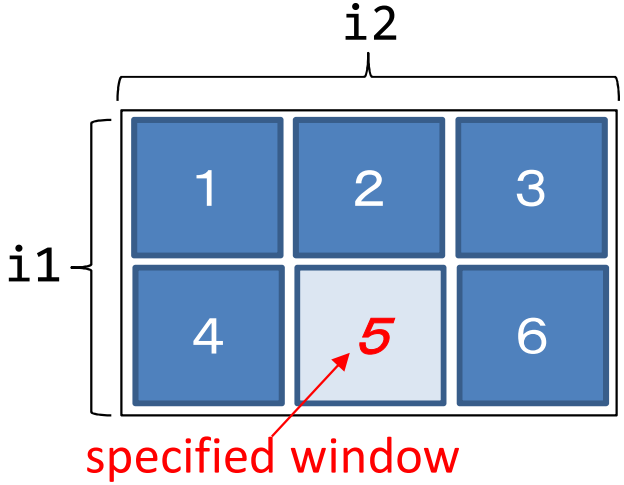
Figure window is a sub-window for graph creation in the graphic window, and each figure window has axis value independent to co-ordinate of the graphic window.

The functions concerned with graph is applied for a figure window

```
void Set_figure(int i1,int i2,int n)
```

Create figure windows with alignment like a matrix consisting of the number columns ‘i1’ and rows ‘i2’

Each figure window is numbered from 1 to i1*i2 as shown in the figure. Left upper position is #1.



Ex.) `Set_figure(2,3,5);`

The number of the current figure is specified by the argument ‘n’.
Once the axis range is fixed, the axis range is used after the second time.

In each figure window, Set_figure() should be called at least once before plot functions are called.

Even if the number of plot area is 1, ‘Set_figure(1,1,1);’ should be called.

How to draw 1D graph

The most easiest way to draw a 1D graph is to call 'Plot1d()' function.

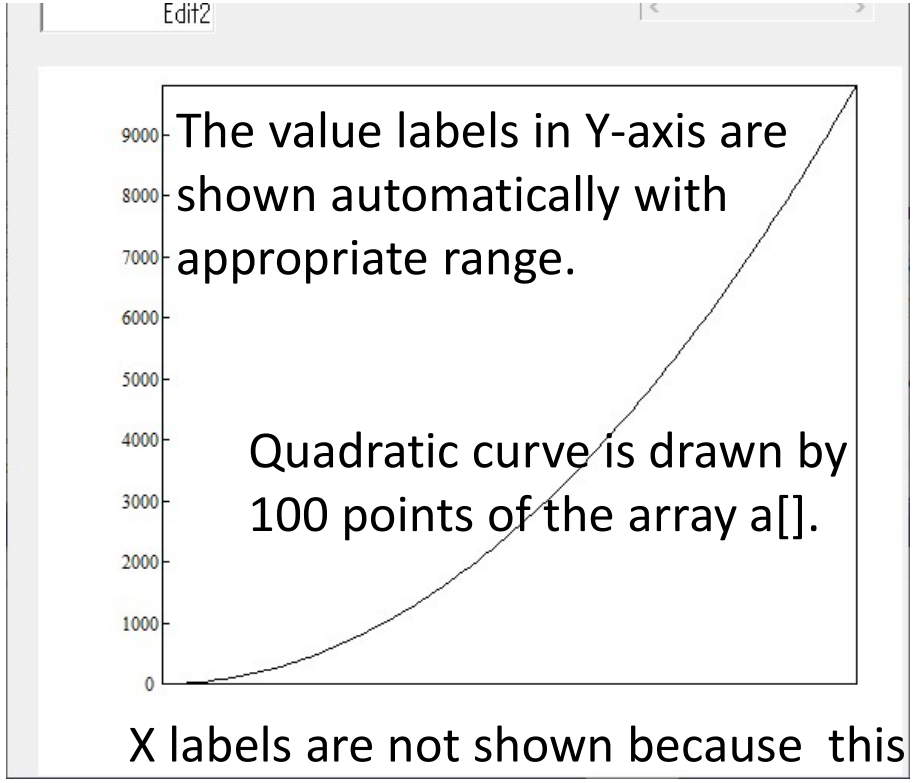
Plot1D function draws a bent line connected with values of array elements specified by a double precision variable array

```
void Plot1d(double *yn ,int n)
```

Ex.) **`Plot1d(x,100);`**

yn: array variable name defined by double precision type
If you draw a line from the element number of 10, it should be specified as &yn[9] .

n : The number of elements to be drawn.



X labels are not shown because this function does not have x-axis.

Example 2

```
#include "def.h"
#include "mlib.h"
int main(int Number) {
  double a[100];
  int i;

  for (i = 0; i < 100; i++) {
    a[i] = i * i;
  }
  Set_figure(1, 1, 1);
  Plot1d(a, 100);
}
```

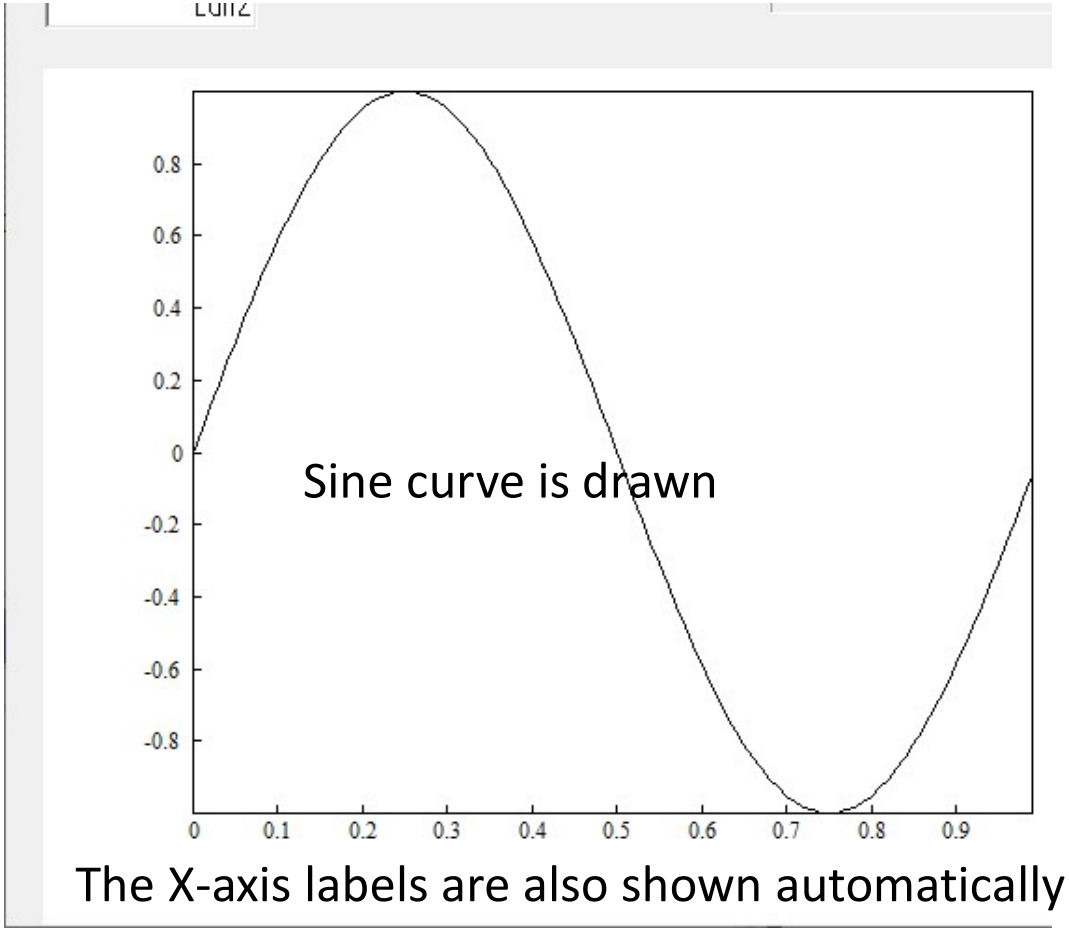
How to draw 1D graph with x-axis

In order to draw 1D graph with x-axis, 'Plotxy()' function is used.

```
void Plotxy(double *xn , double *yn ,int n)
```

Ex.) Plotxy(x,y,100);

xn: array variable name defined by double precision type for horizontal axis.
yn: array variable name defined by double precision type for vertical axis
n : The number of elements to be drawn.



The X-axis labels are also shown automatically

Example 3

```
#include "def.h"
#include "mlib.h"
#define pi 3.141592
int main(int Number) {
  double t[100],y[100];
  int i;
  for (i = 0; i < 100; i++) {
    t[i] = (double)i / 100;
    y[i] = sin(2*pi*t[i]);
  }
  Set_figure(1, 1, 1);
  Plotxy(t,y, 100);
}
```

Setting of Axis

```
void Axis_ycap(double min, double max, char *label)
void Axis_xcap(double min, double max, char *label)
```

Set Y- or X-axis display range and display numerical labels, and captions

min : minimum value to be displayed in Y- or X-axis .
max : maximum value to be displayed in Y- or X-axis .
label : string for caption in Y- or X-axis (less than 100 characters)

Ex.) Axis_ycap(-1,1,"Time [s]"); Axis_xcap(-1,1,"Amplitude [V]");

Plot functions called after setting follows the specified display range.

Font specification

Font used in graphic window can be changed in main program before output.

The font information is defined as global constant as follows.

Available font TIMES, CENTURY, TAHOMA, SYLFAEN, MS GOTHIC, MS PGOTHIC, MS MINCHO, MS PMINCHO, etc

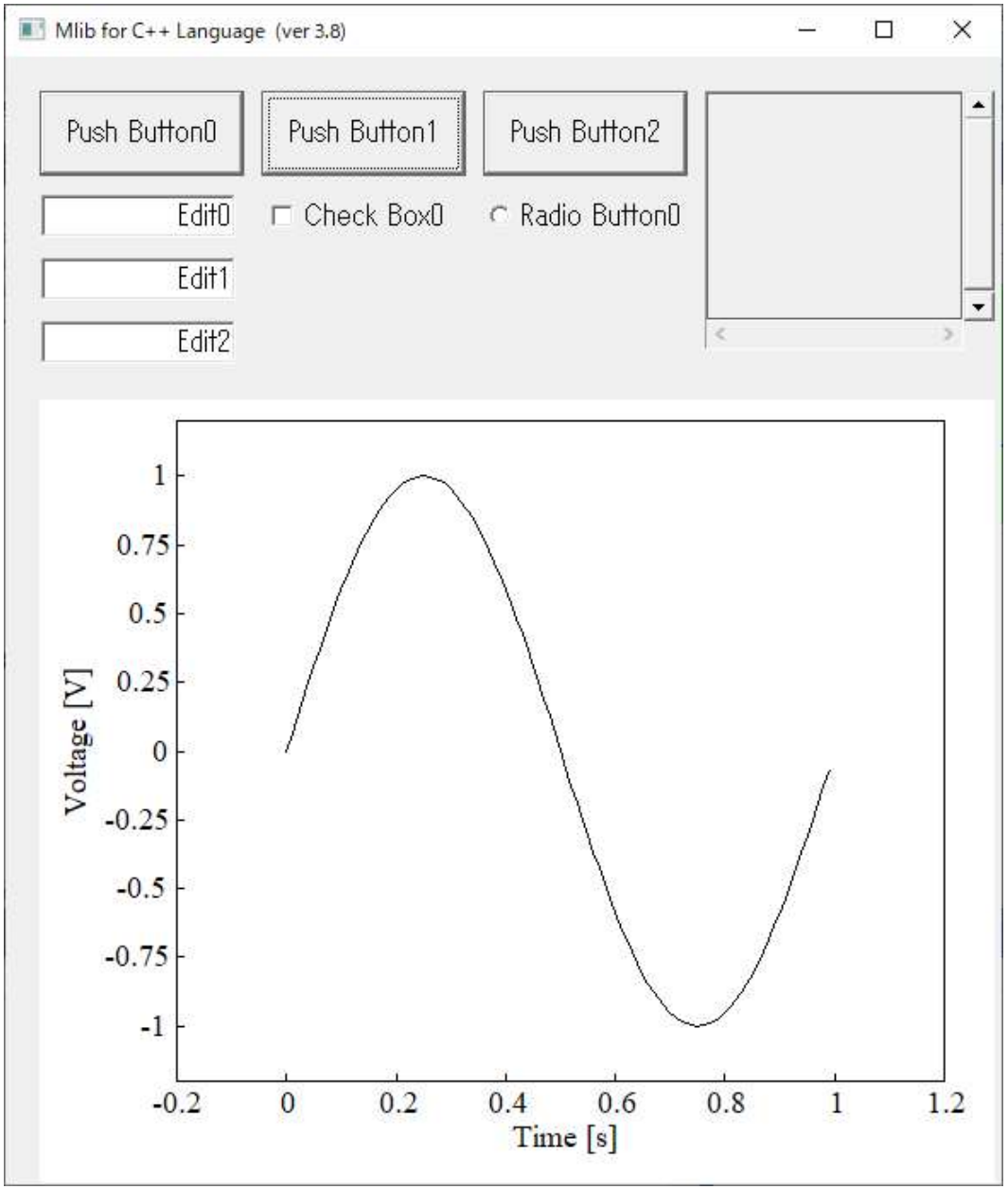
Ex.) lstrcpy(Used_Font,TEXT("MS GOTHIC"));

If you use Italic font, a global variable 'Italic_Font_Flug ' should be 1

The Font size is also specified a global variable 'Used_Font_Size'

Ex.) Used_Font_Size=10;

Example 4




```
#include "def.h"
#include "mlib.h"
#define pi 3.141592

int main(int Number) {
 double t[100],y[100];
 int i;

 for (i = 0; i < 100; i++) {
  t[i] = (double)i / 100;
  y[i] = sin(2*pi*t[i]);
 }

 Set_figure(1, 1, 1);
 Used_Font_Size = 20;
 lstrcpy(Used_Font,TEXT("TIMES"));
 Axis_ycap(-1.2,1.2,"Voltage [V]");
 Axis_xcap(-0.2,1.2,"Time [s]");
 Plotxy(t,y, 100);
}
```

Decoration of drawing (1)

```
void Plot_pen(int pf, int pw, int pc)
```

Ex.) Plot_pen(0,0,1);

Specify the color and line type to draw in the graphics window.

pf :
0 – Solid line
1 – Dash line
2 – Dot line
3 – Dot-chain line
4 – Two-dot-chain line

pw:
The thickness of the pen.
Specify an integer value
in pixel.

pc:
0- Black
1- Red
2- Green
3- Blue
4- Yellow
5- Magenta
6- Cyan
7- White
8- cycle color

Once this function is called, pen setting is kept for subsequent drawing.

```
void Clf(int clflug)
```

Ex.) Clf(1);

Erase figure window or graphic window.

clflug
- -1 Draw only the frame of the current figure window (do not erase)
- 0 Erases the current figure window, and the axis information remains.
- 1 Erases the current figure window including axis information.
- 2 Erase the graphic window including all the window information

Decoration of drawing (2)

```
void Text_draw(double x , double y, char* text);
```

例） `Text_draw(10,200,”sin”);`

Paste the string specified by argument ‘text’ in the figure window.
The position (x, y) is based on the coordinates of each axis in the specified figure window.
Can be used to express information such as graph titles and legends

```
void Grid_on(int grflug)
```

例） `Grid_on(3);`

Draw auxiliary lines on the figure window.
Draw a straight line based on the position with the numeric label.
call after setting the axis information such as maximum / minimum value in each axis. (Plot () function, Axis_ycap (), Axis_xcap ())

grflug
- 0: Draw grid line at only 0 level for y-axis
- 1: Draw grid line only for y-axis
- 2: Draw grid line only for x axis
- 3: Draw grid line for both x-axis and y-axis

Decoration of drawing (3)

```
void Aspect_ratio(double ax, double ay)
```

例） `Aspect_ratio(1,1);`

The default figure window sizes are evenly allocated according to the number of windows determined by the Set_figure() function.
The window size is changed by a specified ratio represented by (width: height) = (ax: ay) within a range that does not exceed the default size.
Must be called before plot function.

```
void Legend(char* text, int posflug)
```

例） `Legend("a|b|c",4);`

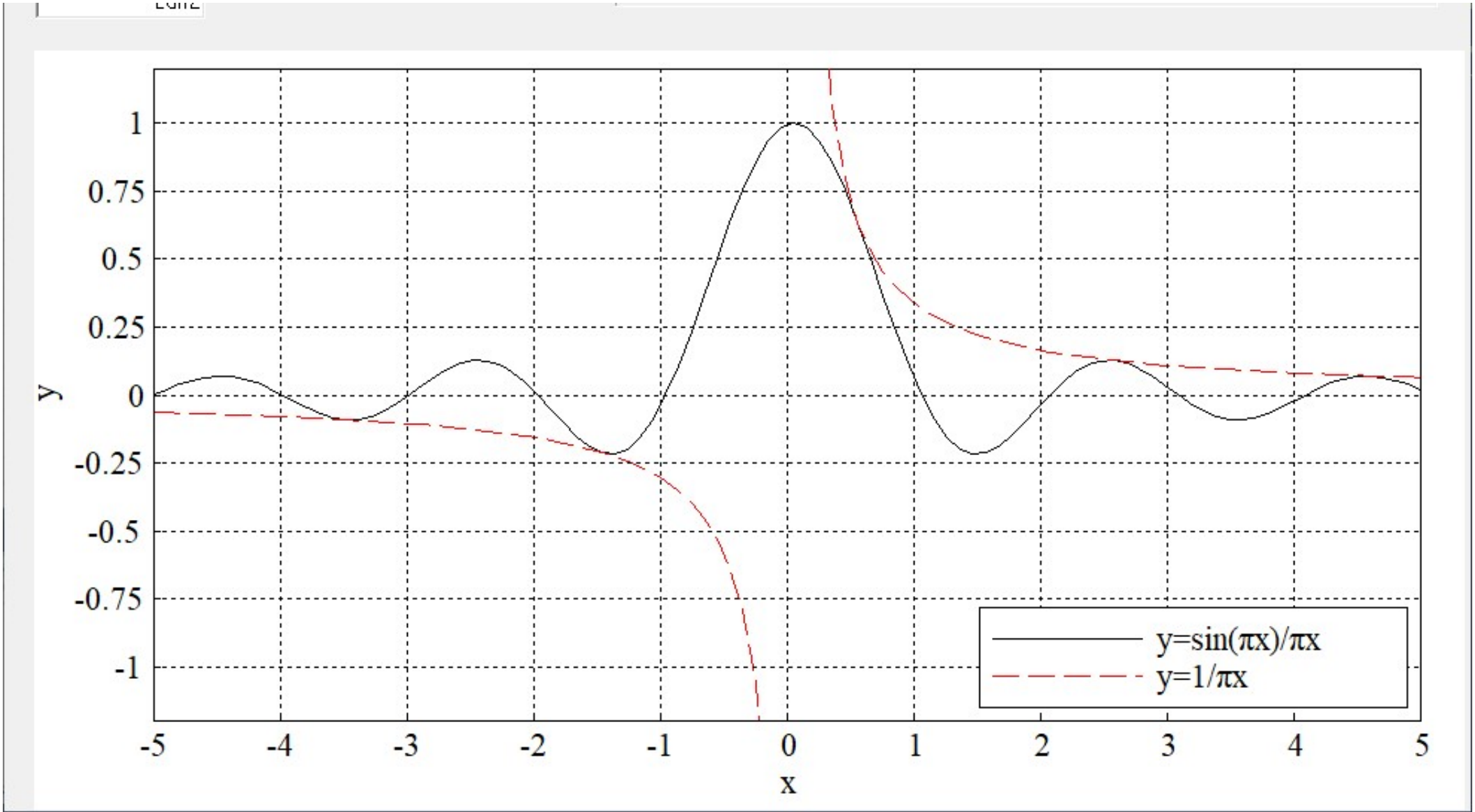
If you draw continuously by Plot () function, it will be overwritten.
Legend() function displays legends for the number of the overlaid graphs.
The string of the legend should be written as "line1 | line2 | line3". The name of each line is delimited the character '|' in the string .

the legend position is specified by `posflug` as

- 0 : at the bottom right outside the frame
- 1 : at the upper left corner of the frame
- 2 : at the upper right corner of the frame
- 3 : at the lower left corner of the frame
- 4 : at the lower right corner of the frame

Example 5

Display the Sinc function as y = sin (πx) / πx in the range of $-5 \leq x \leq 5$ with x-axis and y-axis labels. In addition, overwrite y = 1 / πx in red and dashed lines. and display the legend.



Program of Example 5

```
#define N 100
#define pi 3.1415926535
int main (int Number){
   int i;
   double x[N],y[N];

   for(i=0;i<N;i++){
      x[i]=10.0/N*i-5.0;
      y[i]=sin(pi*x[i])/(pi*x[i]);
   }
   y[N/2]=1;
   Set_figure(1,1,1);
   Used_Font_Size=45;
   Axis_xcap(-5,5,"x");
   Axis_ycap(-1.2,1.2,"y");
   Plotxy(x,y,N);

   for(i=0;i<N;i++){
     y[i]=1.0/(pi*x[i]);
   }
   Plot_pen(1,1,1);
   Plotxy(x,y,N);
   Grid_on(3);
   Legend("y=sin(πx)/πx|y=1/πx",4);
}
```

Constant setting

Sin function needs ‘math.h’. But it has already be included in ‘def.h’

Setting array data x (-5<x<5) .

Setting array data y=sin(πx)/ πx using array x.

Caution. When $x = 0$, $\sin(x)/x$ is Infinity. So, directly specified.

Setting x-axis range and caption.

Setting y-axis range and caption.

plot data y with x.

Setting array data y=1/ πx using array x.

Change pen condition to red color and dashed line.

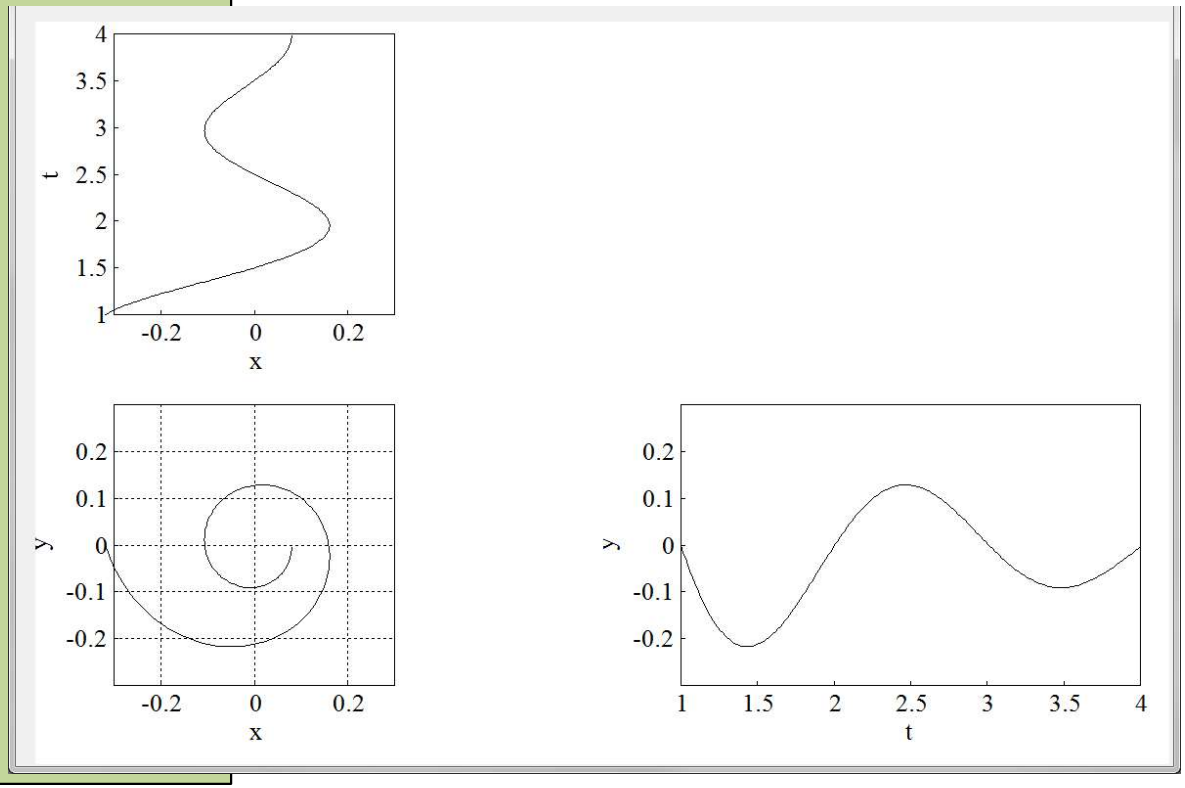
Overwrite a graph

Add grid line

Insert legend of the graphs

Example 6

```
int main (int Number){
    int i;
    double pi=3.1415926535;
    double x[N],y[N],t[N];
    for(i=0;i<=N;i++){
        t[i]=3.0/N*i+1;
        x[i]=cos(pi*t[i])/(pi*t[i]);
        y[i]=sin(pi*t[i])/(pi*t[i]);
    }
    Used_Font_Size=30;
    Set_figure(2,2,4);
    Axis_xcap(1,4,"t");
    Axis_ycap(-0.3,0.3,"y");
    Plot1d(y,N);
    Set_figure(2,2,1);
    Aspect_ratio(1,1);
    Axis_xcap(-0.3,0.3,"x");
    Axis_ycap(1,4,"t");
    Plotxy(x,t,N);
    Set_figure(2,2,3);
    Aspect_ratio(1,1);
    Axis_xcap(-0.3,0.3,"x");
    Axis_ycap(-0.3,0.3,"y");
    Plotxy(x,y,N);
    Grid_on(3);
}
```

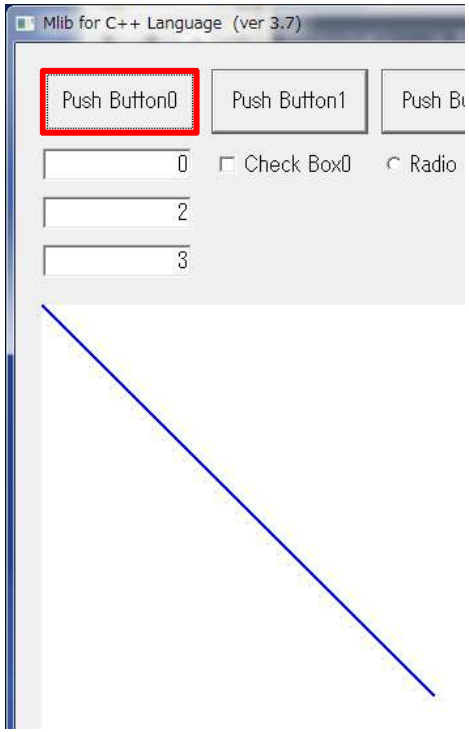


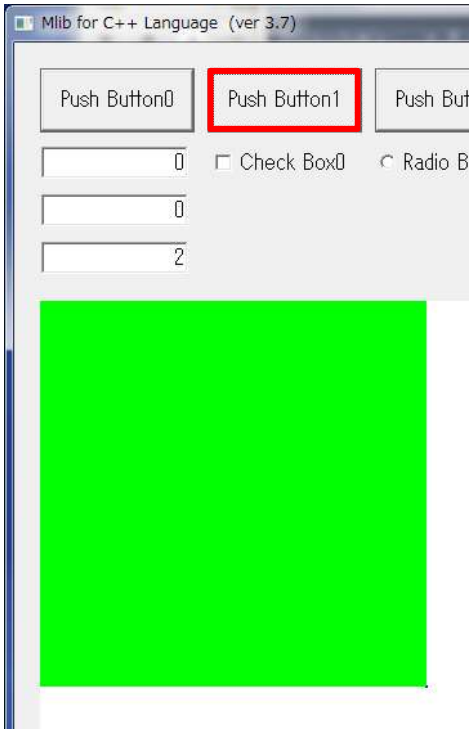
Example 7

Create programs to draw straight lines and squares corresponding to push buttons 0 and 1, respectively.

The line style, thickness and color are specified by the integer values entered in edit boxes 0, 1, and 2 which correspond to the three arguments of the Plot_pen () function.

Push button 2 fills the square area with white color for erasing the drawings.






```
int main (int Number){
   int a,b,c;
   a=Get_int(0);
   b=Get_int(1);
   c=Get_int(2);

   Plot_pen(a,b,c);

   if (Number==0){
      Line(0,0,300,300);

   } else if (Number==1){
      Rect(0,0,300,300,1);

   } else {
      Plot_pen(0,1,7);
      Rect(0,0,300,300,1);
   }
}
```

How to move drawings (animation)

The simple way to animate a figure is to repeat a process of drawing and erase.

In this procedure, 'refresh of window' is important. All the drawing functions is drawn not directly in the display but in the memory of PC. The graphic data in the memory is displayed only after the main function is finished.

So, when you want to move drawing, the memory data in end of each repetition should forced to be displayed. This operation is done by a 'refresh()' function as 'refresh of window'.

The variable 'j' is a counter of repetition in which sine wave data is created, erased and plotting and window is refreshed.

```
#include "def.h"
#include "mlib.h"
#define pi 3.141592
#define N 1000
int main(int Number) {
  double t[N], y[N];
  int i,j;
  Used_Font_Size = 20;
  lstrcpy(Used_Font, TEXT("TIMES"));
  for (j=0;j<100;j++){
    for (i = 0; i < N; i++) {
      t[i] = (double)i / N;
      y[i] = sin(2 * pi *j* t[i]);
    }
    Clf(2);
    Set_figure(1, 1, 1);
    Axis_ycap(-1.2, 1.2, "Voltage [V]");
    Axis_xcap(-0.2, 1.2, "Time [s]");
    Plotxy(t, y, N);
    Refresh();
  }
}
```

Frequency changes by the variable j

Erase figure

Plot figure

Refresh of window

Example 8