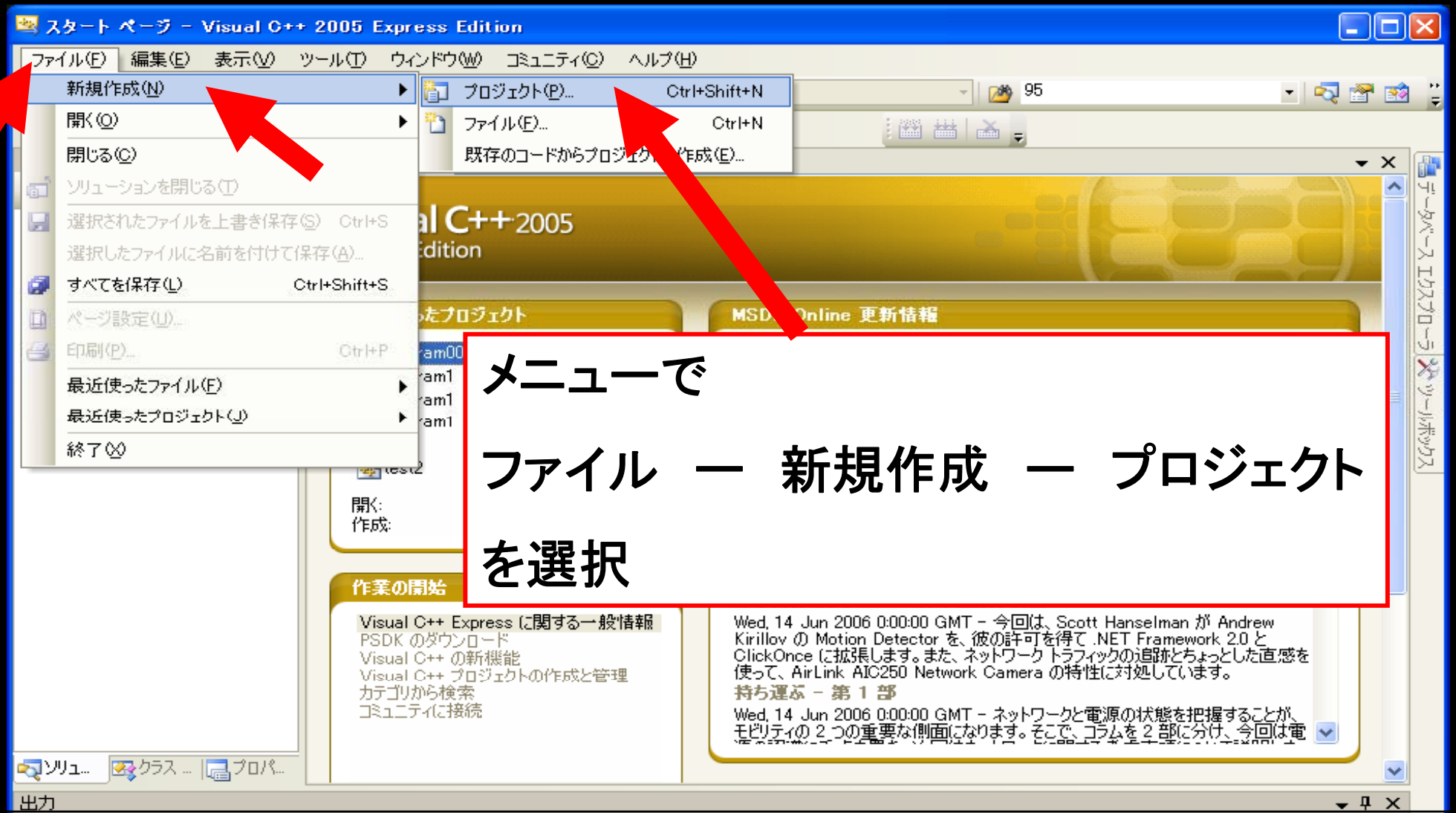
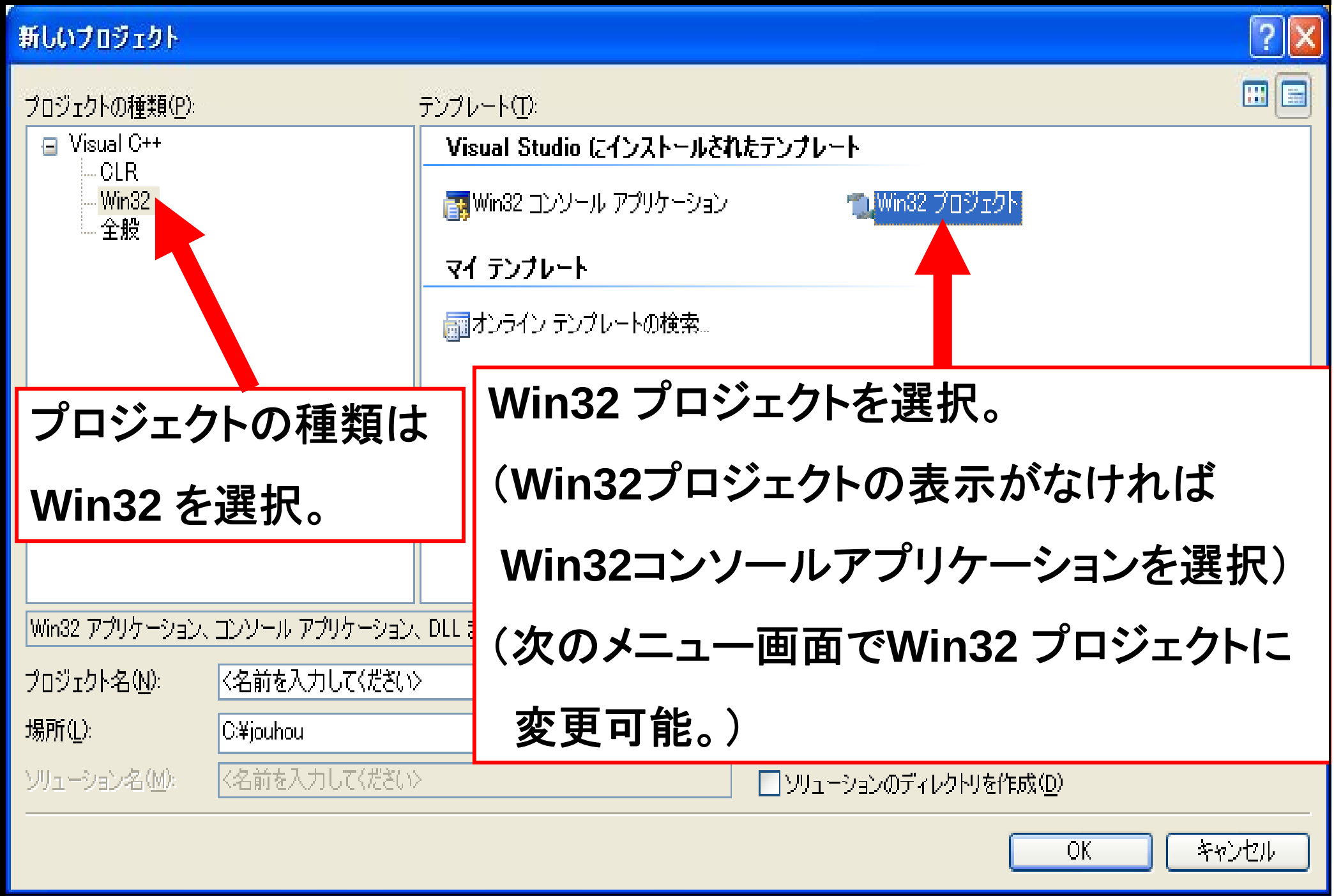
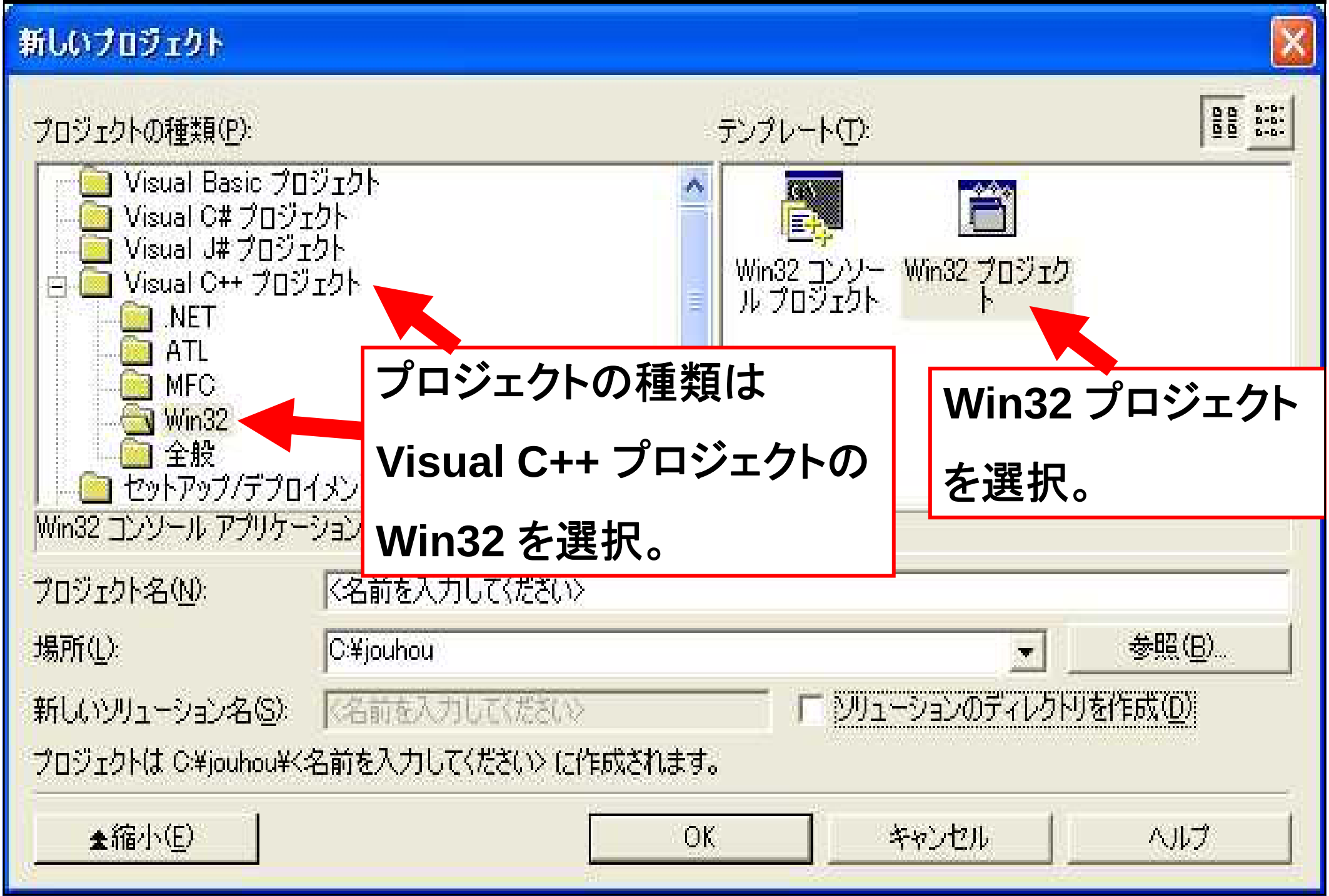
Visual C++ 2005 Express または 端末室PCの Visual Studio .Net で 既存のCプログラムを利用して新たなCプログラムを作る。（前回のprogram0 から program1 を作成する）



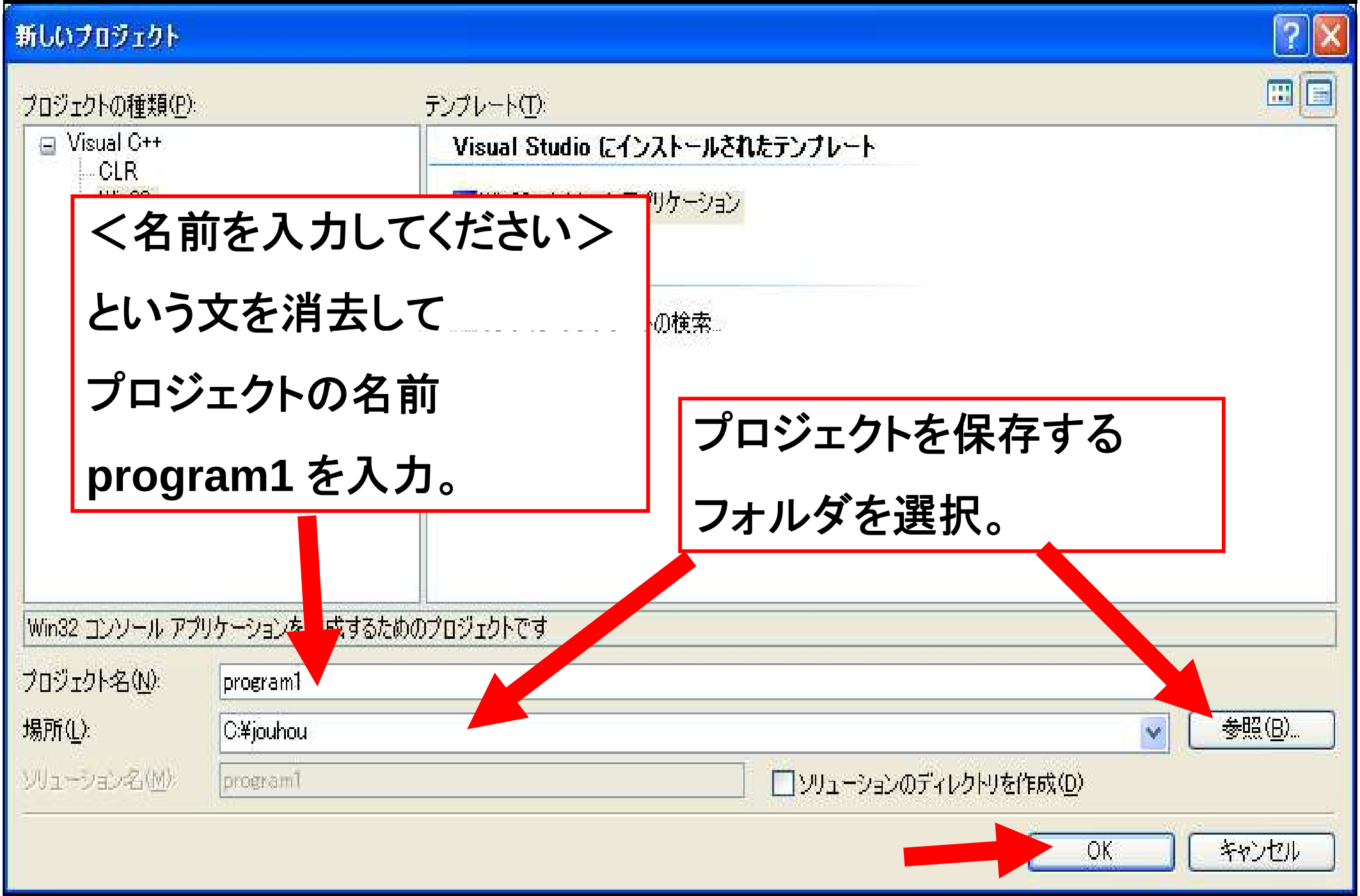
新しいプロジェクトを作成するダイアログが現れる。

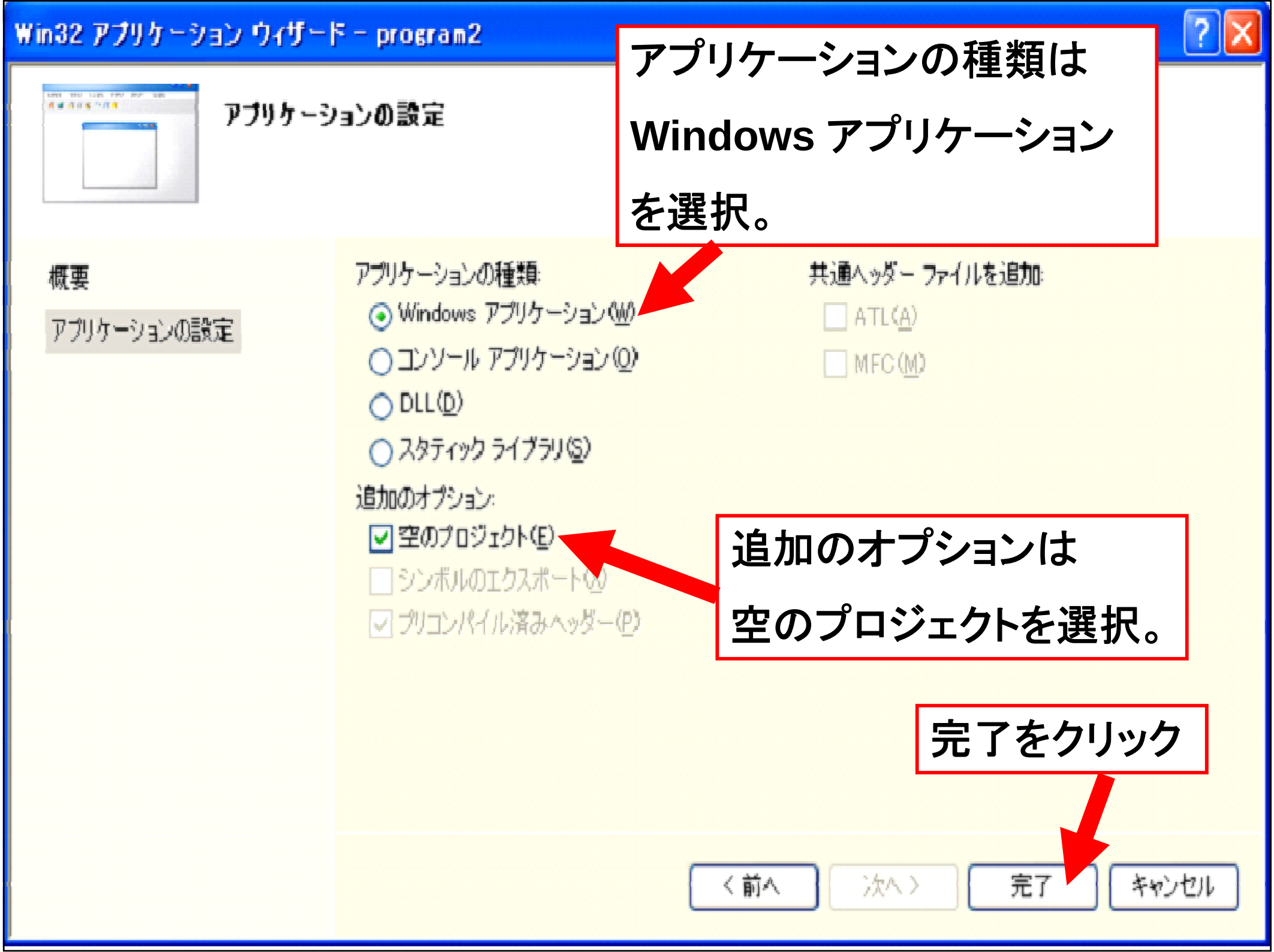


Visual Studio .Net の場合

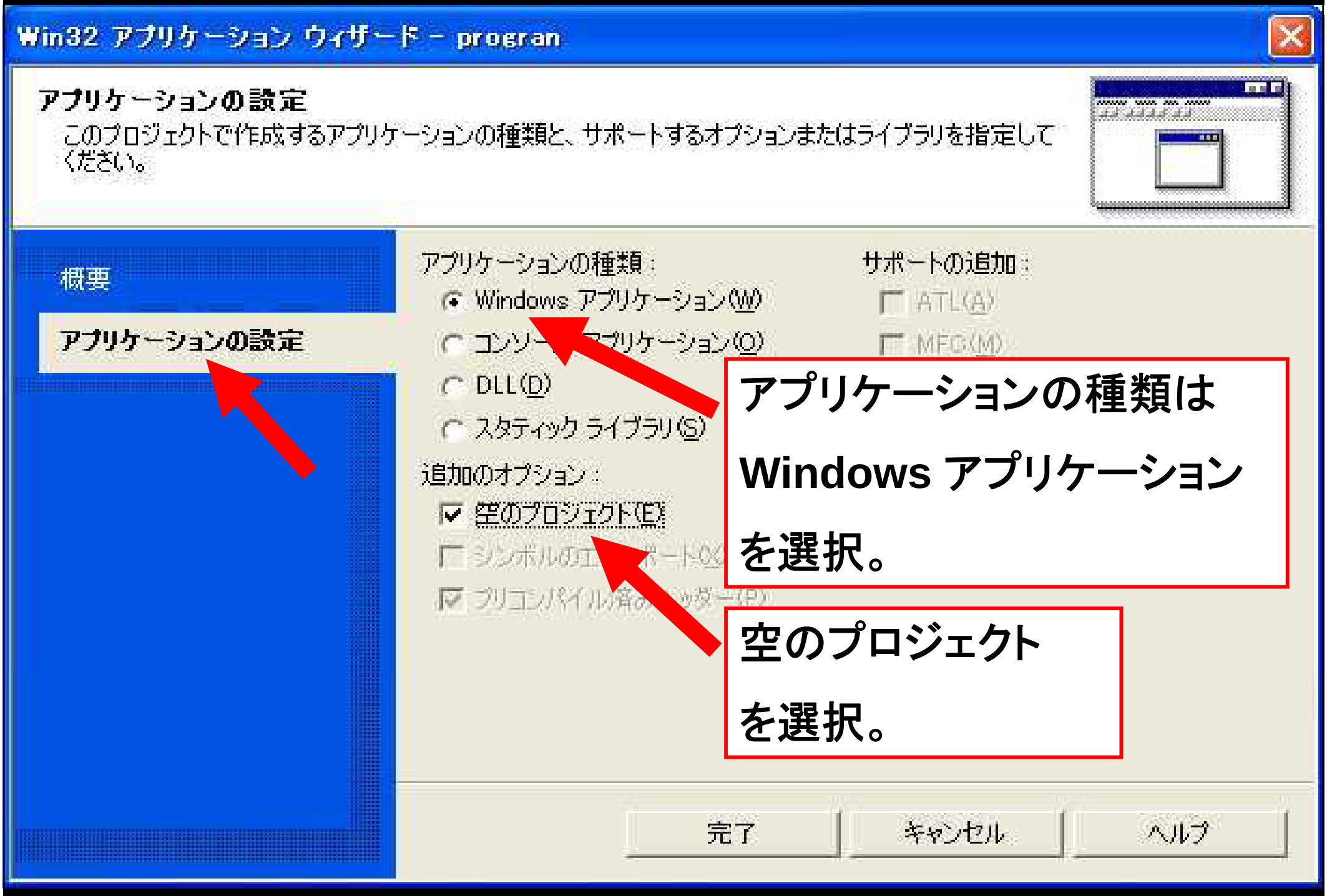


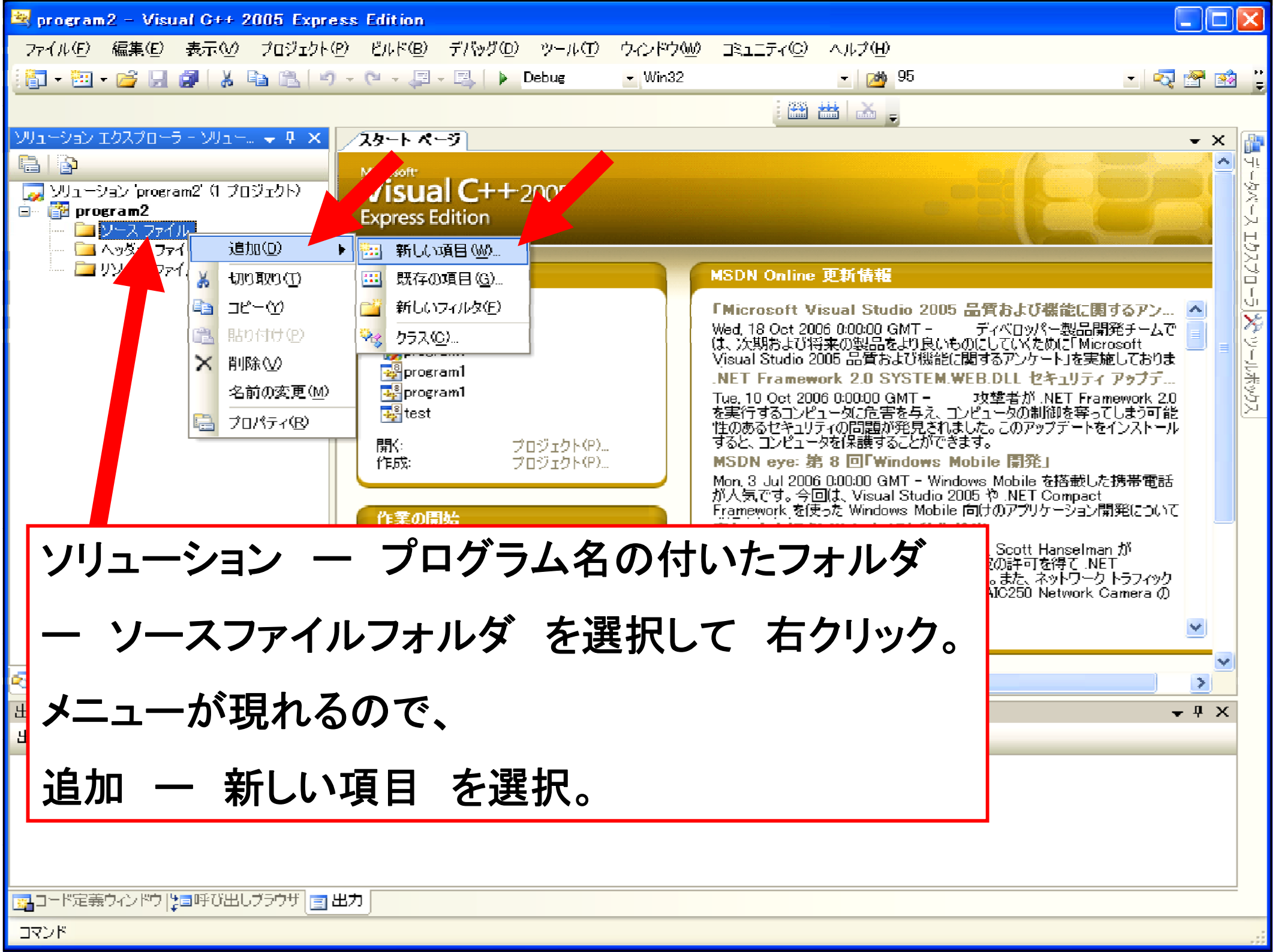
新しいプロジェクトの名前と保存ディレクトリを入力。




Win32 アプリケーション ウィザード - program2
アプリケーションの設定
アプリケーションの種類は
Windows アプリケーション
を選択。
概要
アプリケーションの設定
アプリケーションの種類:
Windows アプリケーション(W)
コンソール アプリケーション(O)
DLL(D)
スタティック ライブラリ(S)
共通ヘッダー ファイルを追加:
ATL(A)
MFC(M)
追加のオプション:
空のプロジェクト(E)
シンボルのエクスポート(X)
プリコンパイル済みヘッダー(P)
追加のオプションは
空のプロジェクトを選択。
完了をクリック
< 前へ
次へ >
完了
キャンセル

Visual Studio .Net の場合





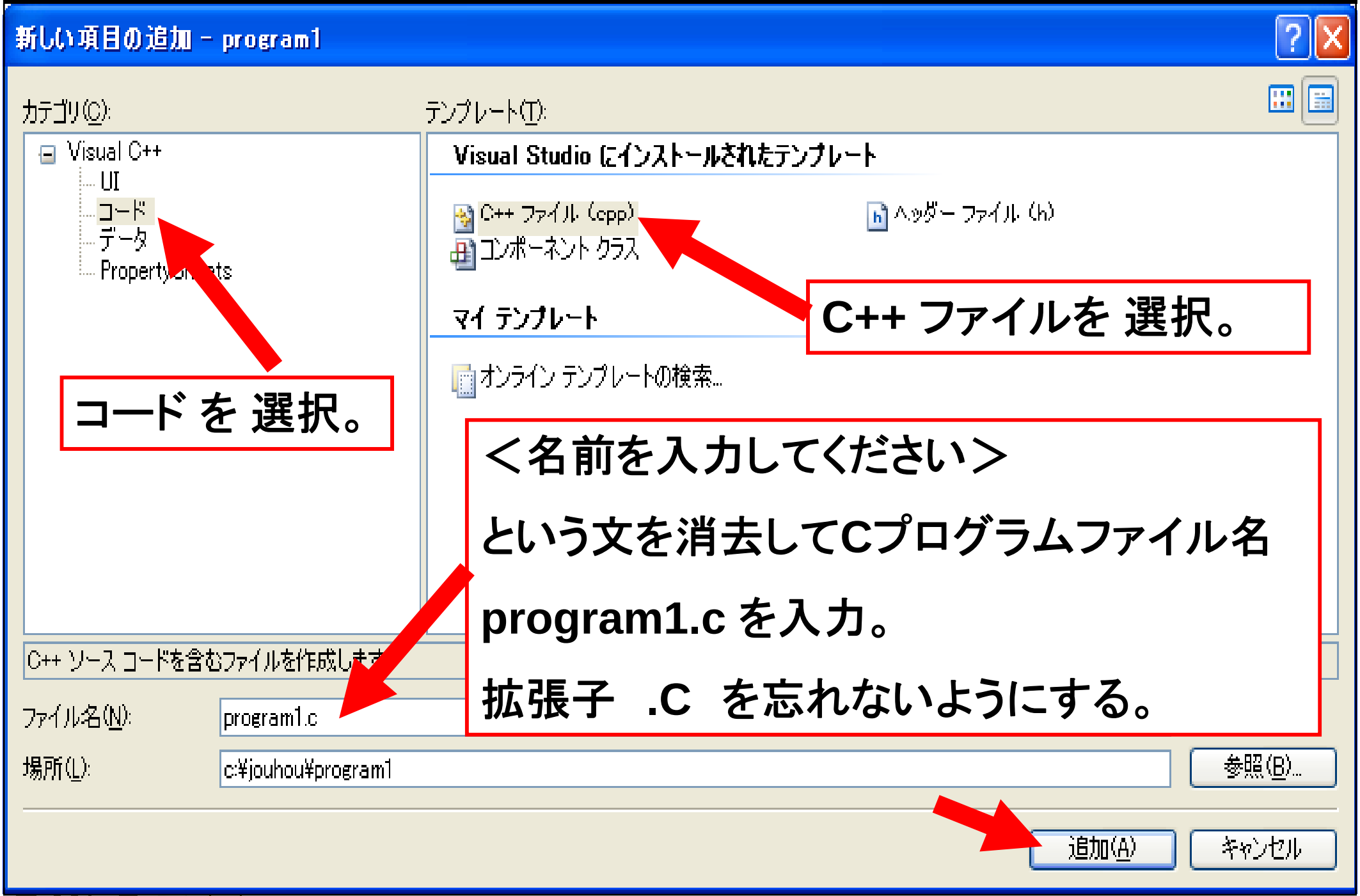
ソリューション　ー　プログラム名の付いたフォルダ

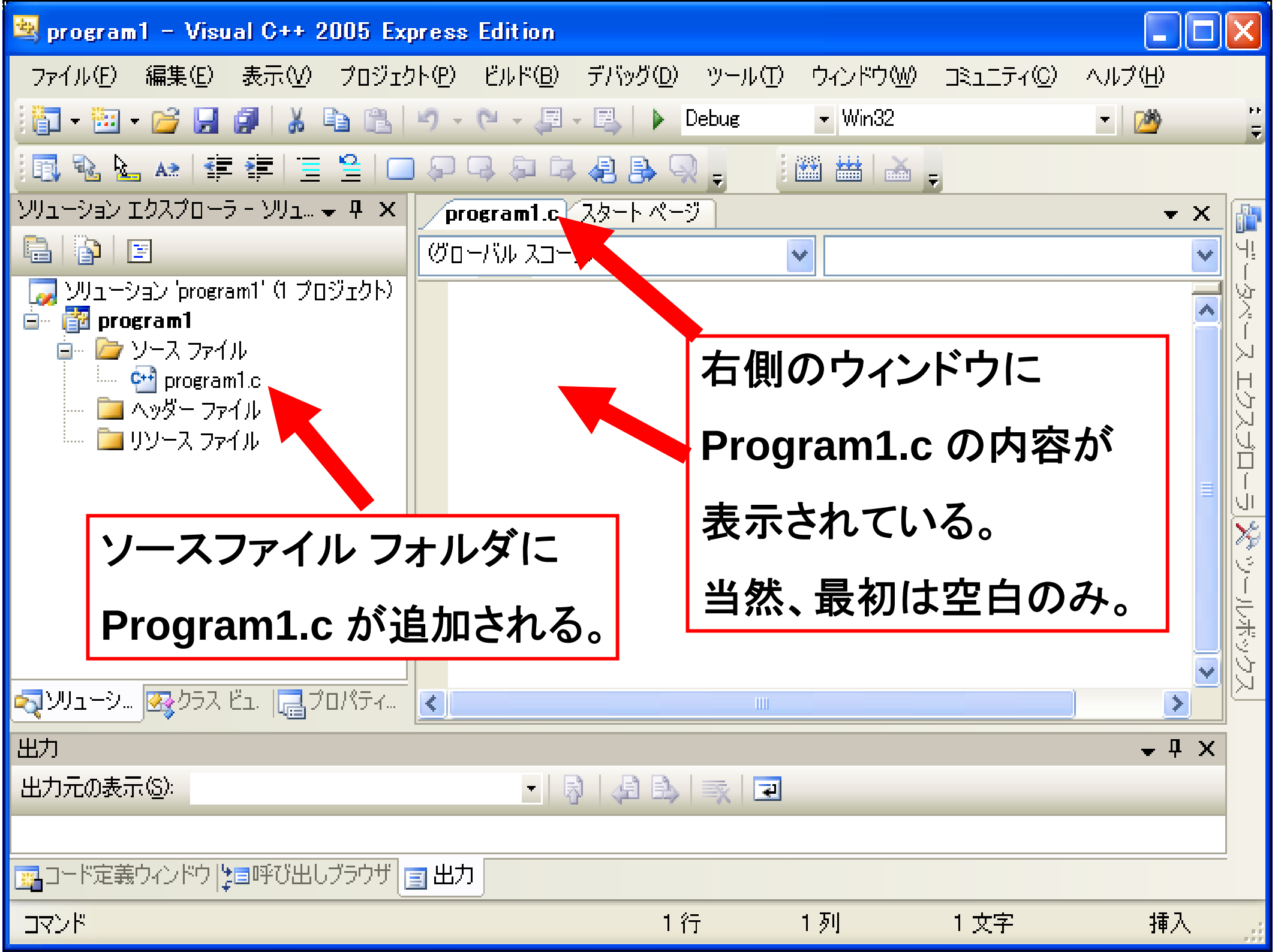
ー　ソースファイルフォルダ　を選択して　右クリック。

メニューが現れるので、

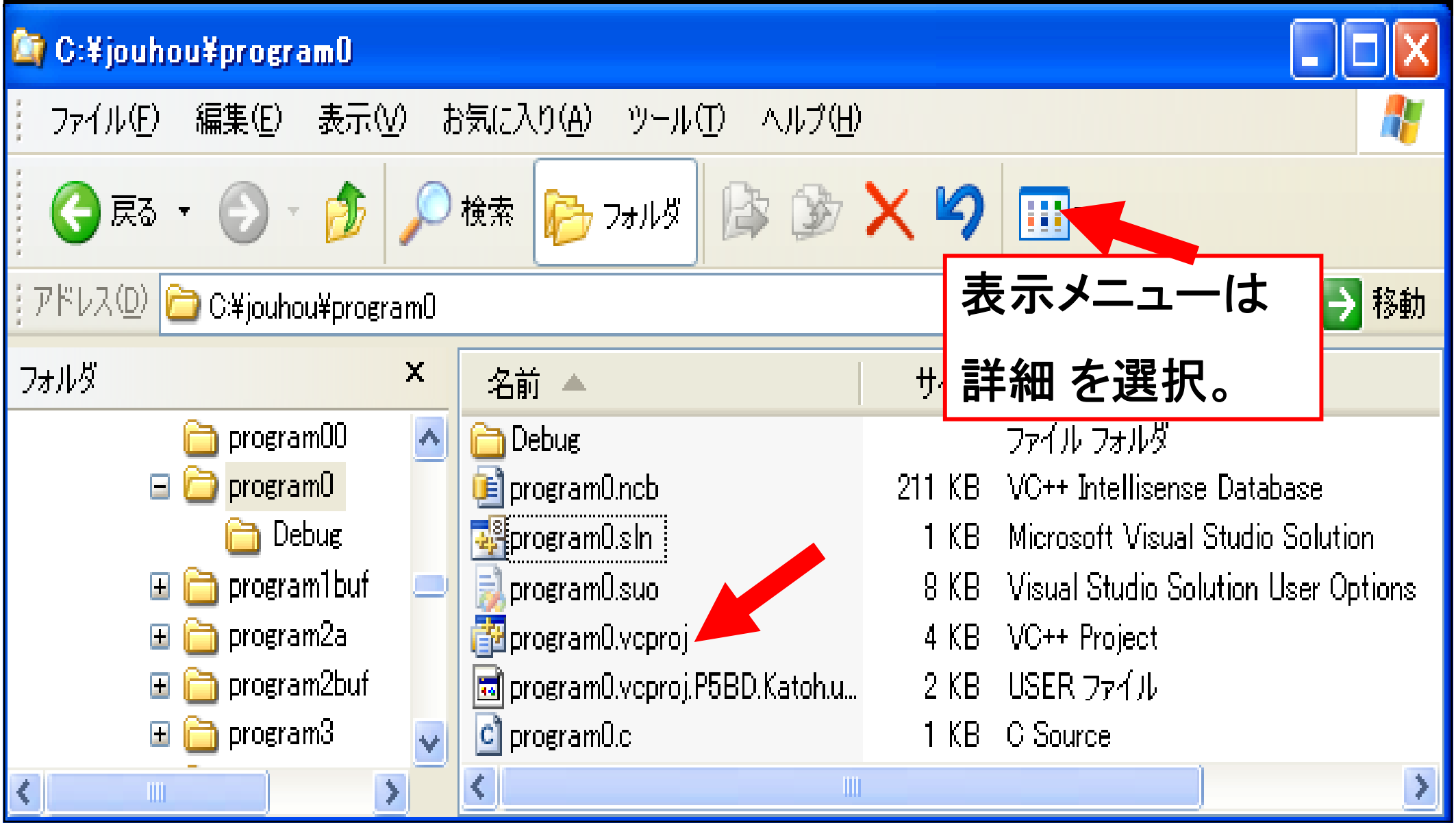
追加　ー　新しい項目　を選択。

新しい項目の追加 ダイアログ が現れる。

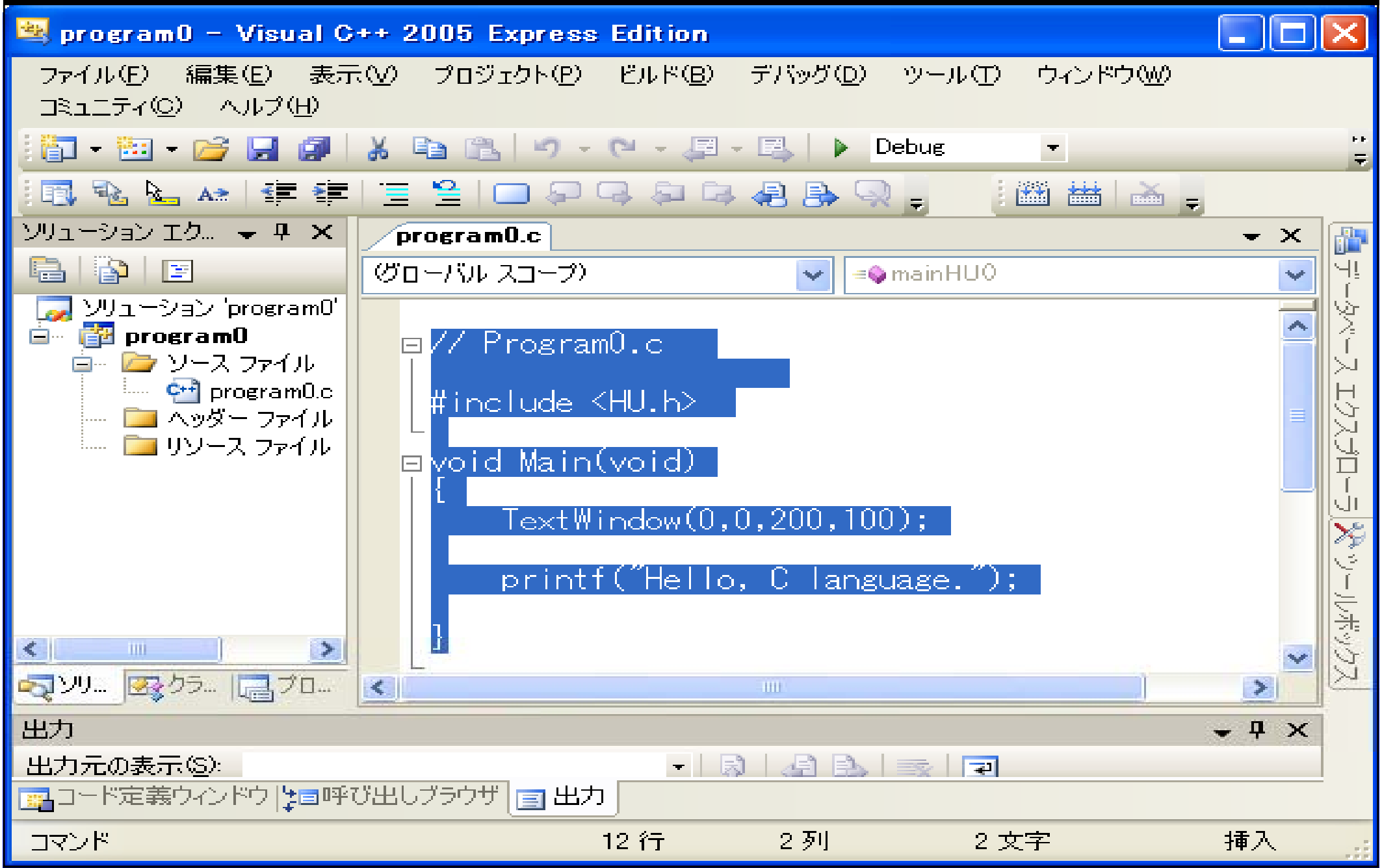



program1 - Visual C++ 2005 Express Edition
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) ツール(T) ウィンドウ(W) コミュニティ(C) ヘルプ(H)
Debug
Win32
ソリューション エクスプローラ - ソリュ...
ソリューション 'program1' (1 プロジェクト)
program1
ソース ファイル
program1.c
ヘッダー ファイル
リソース ファイル
program1.c
スタート ページ
(グローバル スコープ)
ソリューシ...
クラス ビュ...
プロパティ...
出力
出力元の表示(S):
コード定義ウィンドウ
呼び出しブラウザ
出力
コマンド
1 行
1 列
1 文字
挿入
データベース エクスプローラ
ツールボックス
ソースファイル フォルダに
Program1.c が追加される。
右側のウィンドウに
Program1.c の内容が
表示されている。
当然、最初は空白のみ。

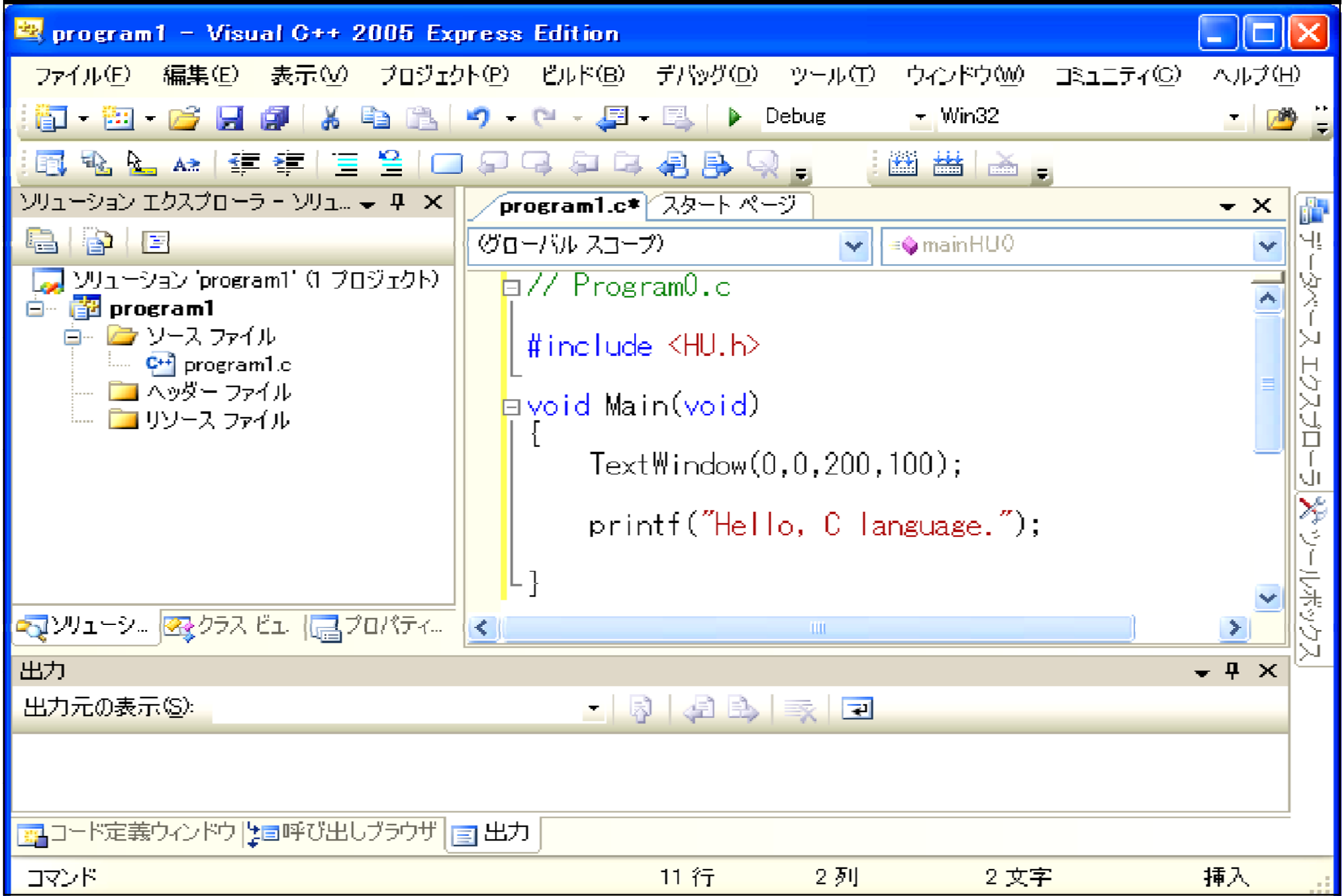
program0.c の内容をコピーする。
前回作成したprogram0 フォルダ内の
program0.vcproj ファイルをダブルクリックする。



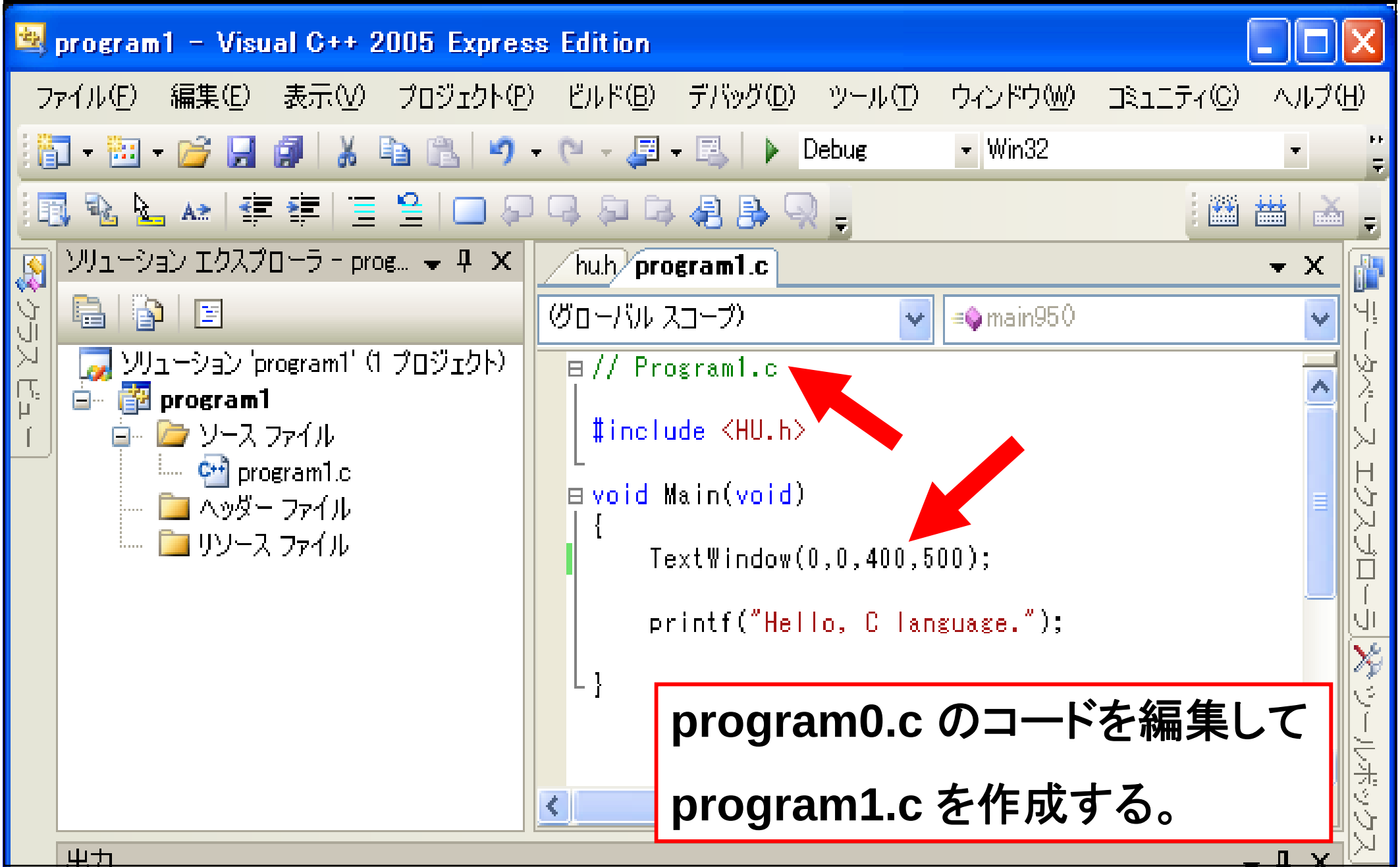
program0.c の内容が表示されるので
全文を選択してコピーする。

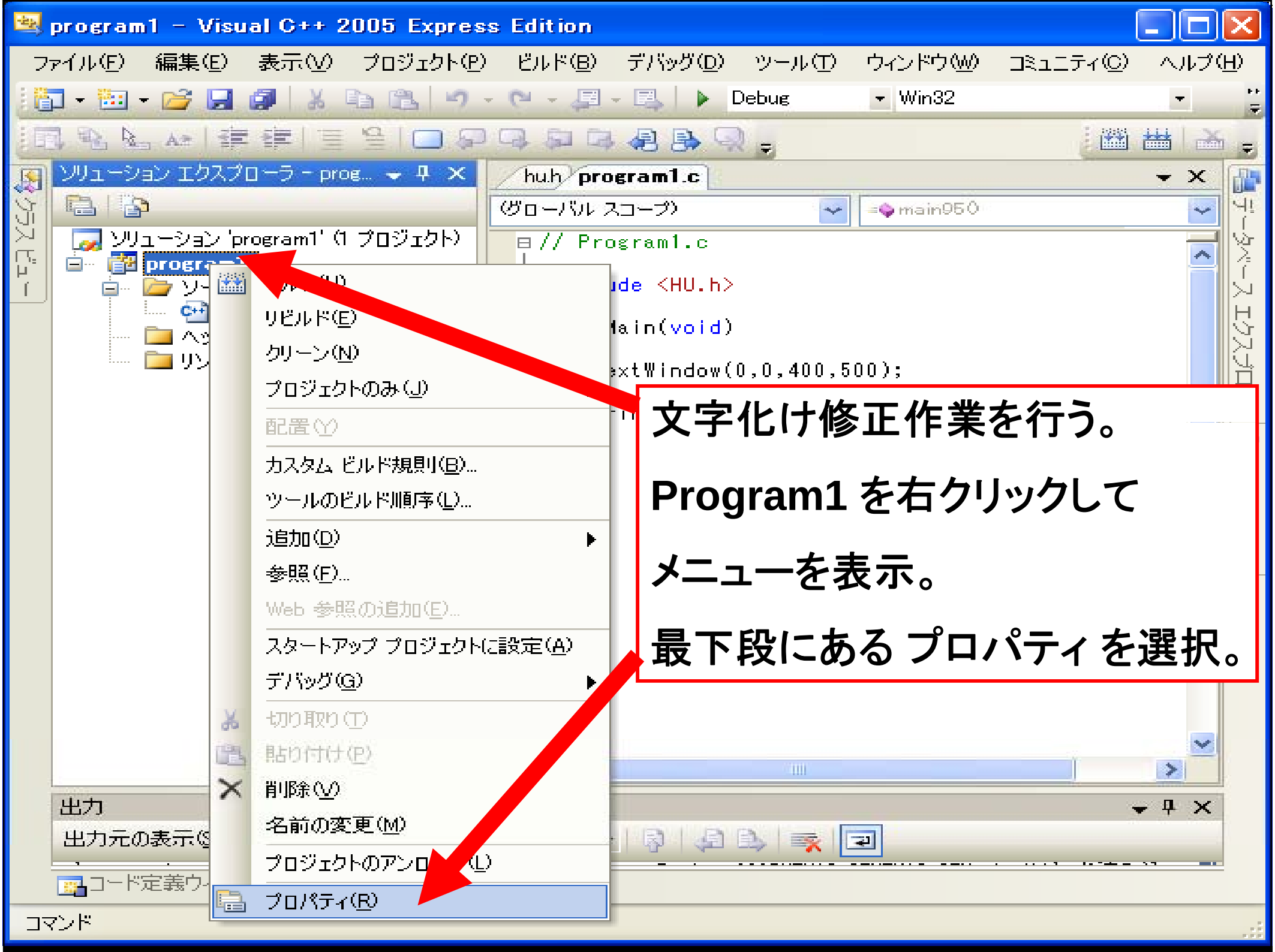


program1.c にprogram0.c をペーストする。

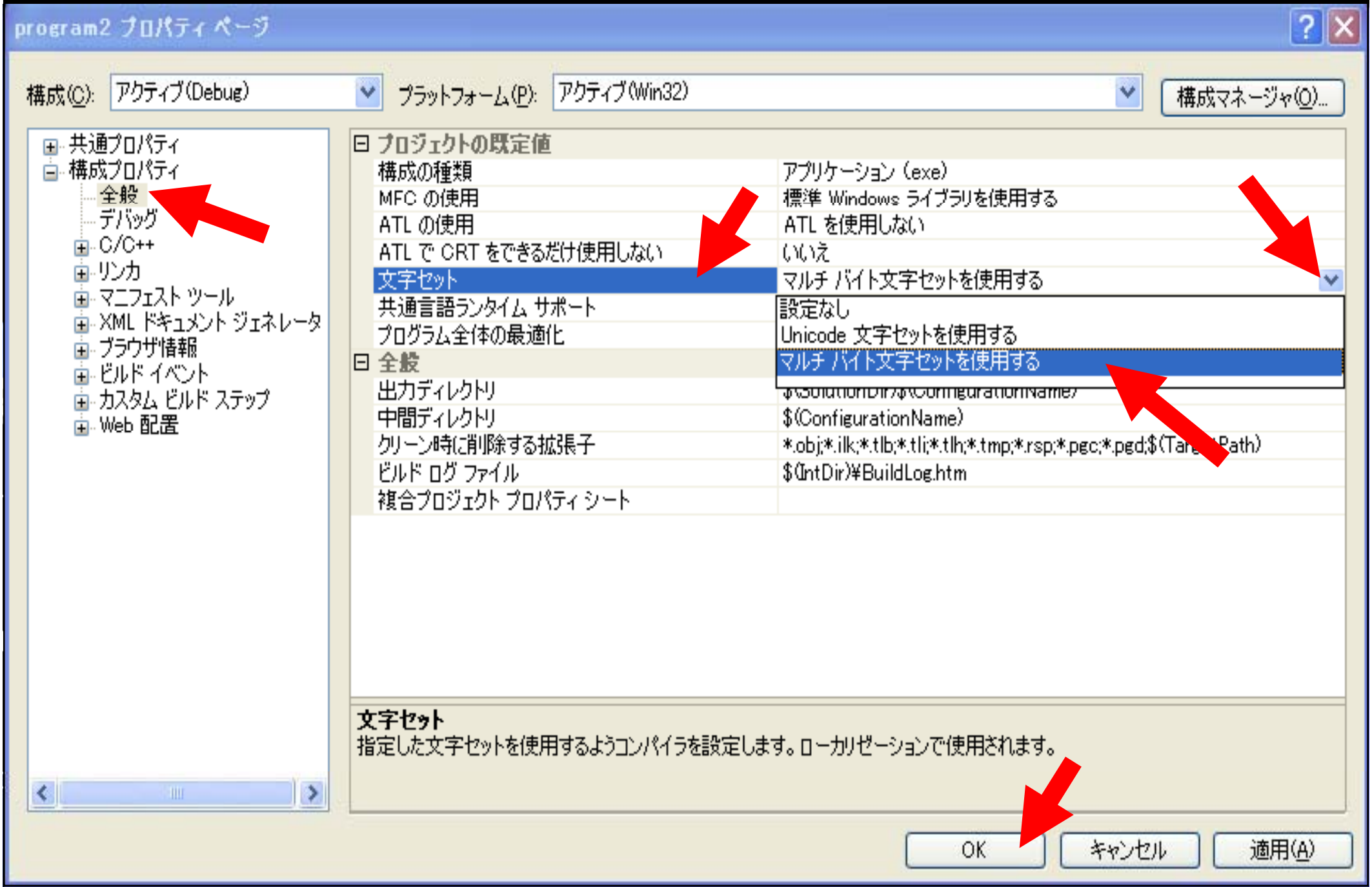


プログラム開発は、既に作成したものを編集、改良して作成する作業が通常の方法。



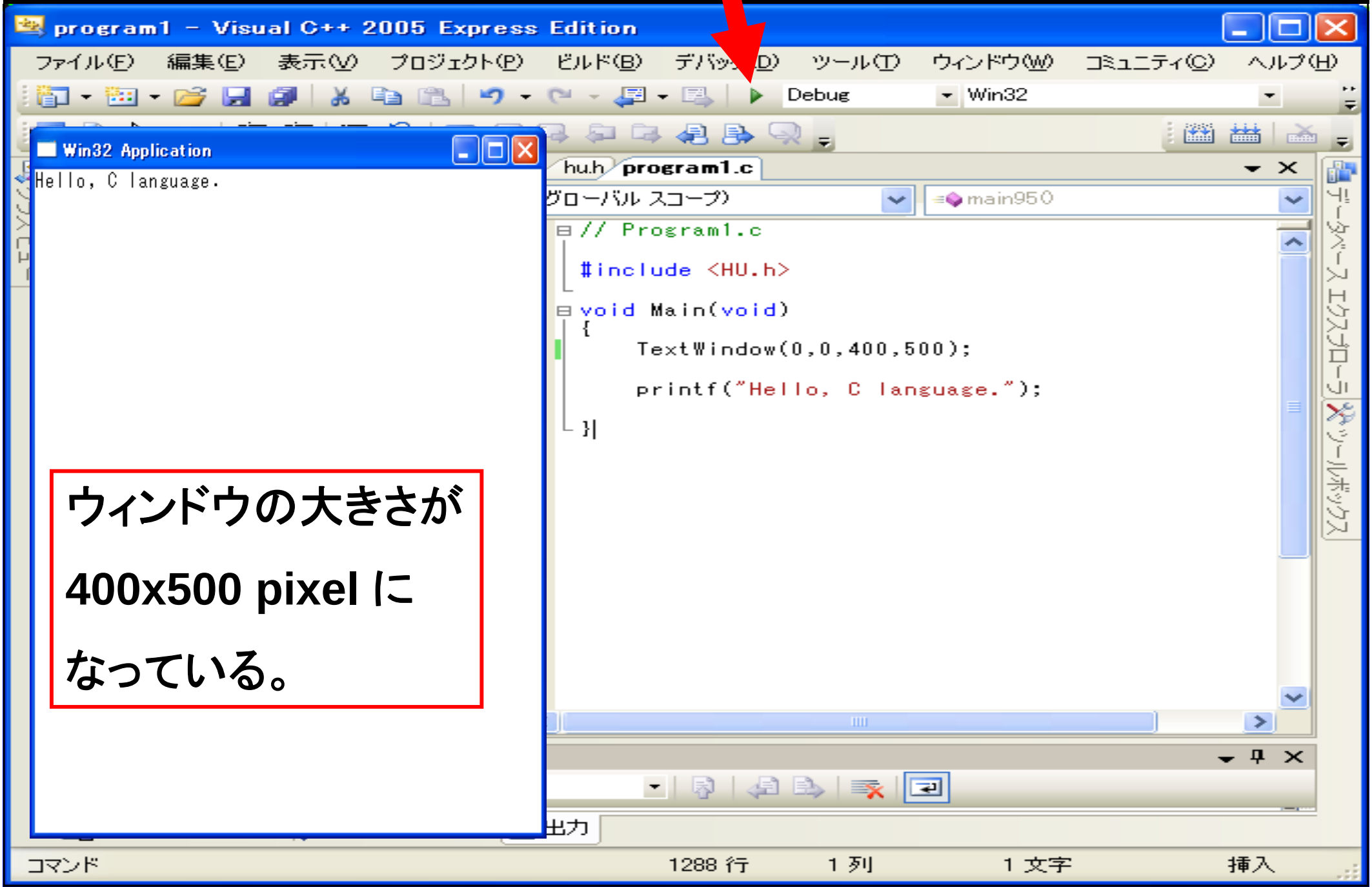

program1 - Visual C++ 2005 Express Edition
ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) ツール(T) ウィンドウ(W) コミュニティ(C) ヘルプ(H)
Debug
Win32
ソリューション エクスプローラ - prog...
ソリューション 'program1' (1 プロジェクト)
hu.h
program1.c
(グローバル スコープ)
main950
// Program1.c
ude <HU.h>
ain(void)
extWindow(0,0,400,500);
リビルド(E)
クリーン(N)
プロジェクトのみ(J)
配置(Y)
カスタム ビルド規則(B)...
ツールのビルド順序(L)...
追加(D)
参照(F)...
Web 参照の追加(E)...
スタートアップ プロジェクトに設定(A)
デバッグ(G)
切り取り(T)
貼り付け(P)
削除(V)
名前の変更(M)
プロジェクトのアンロ (L)
プロパティ(R)
出力
出力元の表示(S)
コード定義ウ
コマンド
文字化け修正作業を行う。
Program1 を右クリックして
メニューを表示。
最下段にある プロパティ を選択。

構成プロパティ　ー　全般 を選択。　文字セットを変更する。
Unicode文字セット になっているので、マルチバイト文字セットに変更。



program1 の実行。

実行ボタン（緑三角印）をクリック。



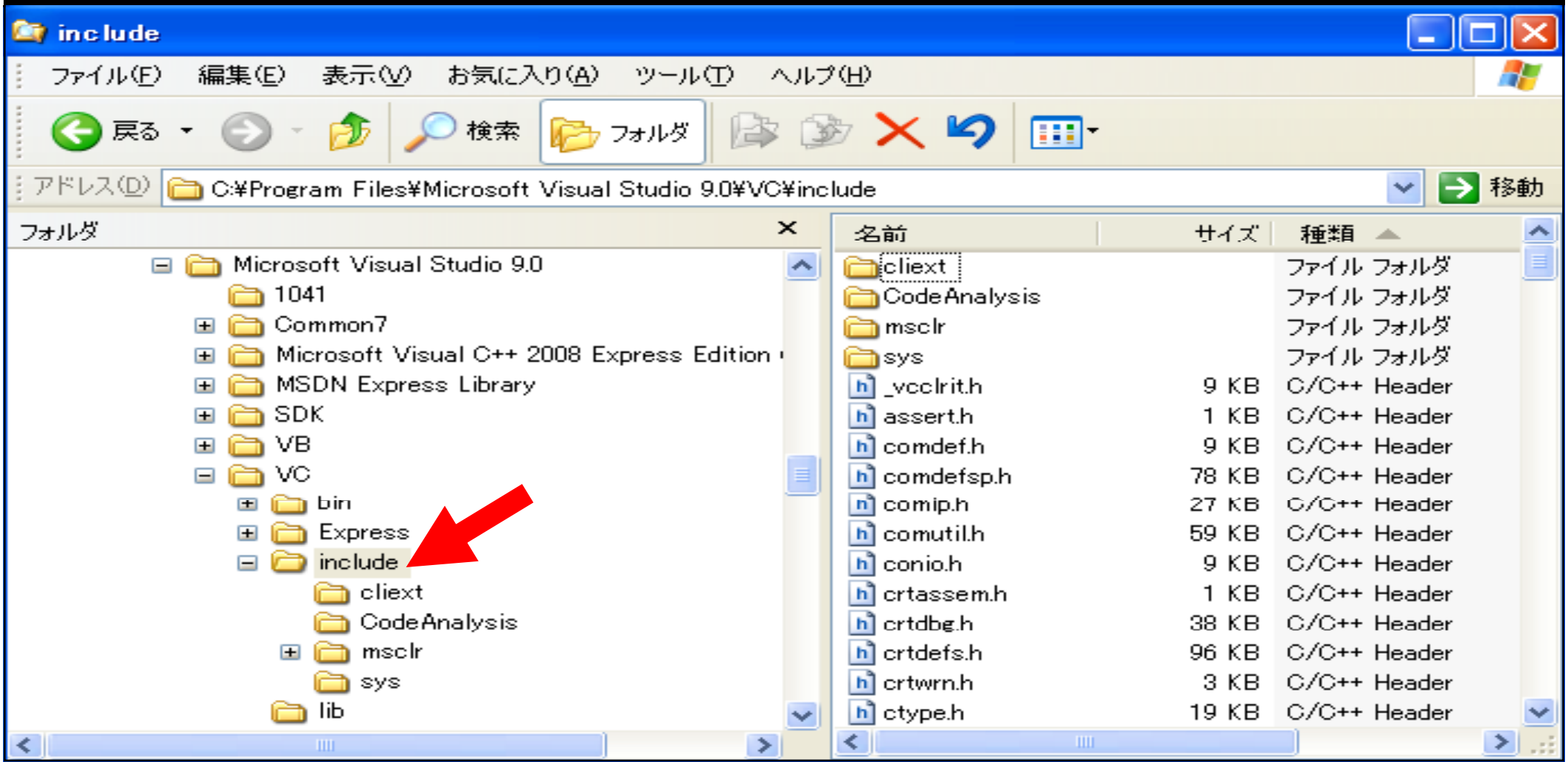
端末室のPCでprogram1 を実行する場合は、
program1 のプロジェクトフォルダ内に
ヘッダファイル HU.h をコピーして、

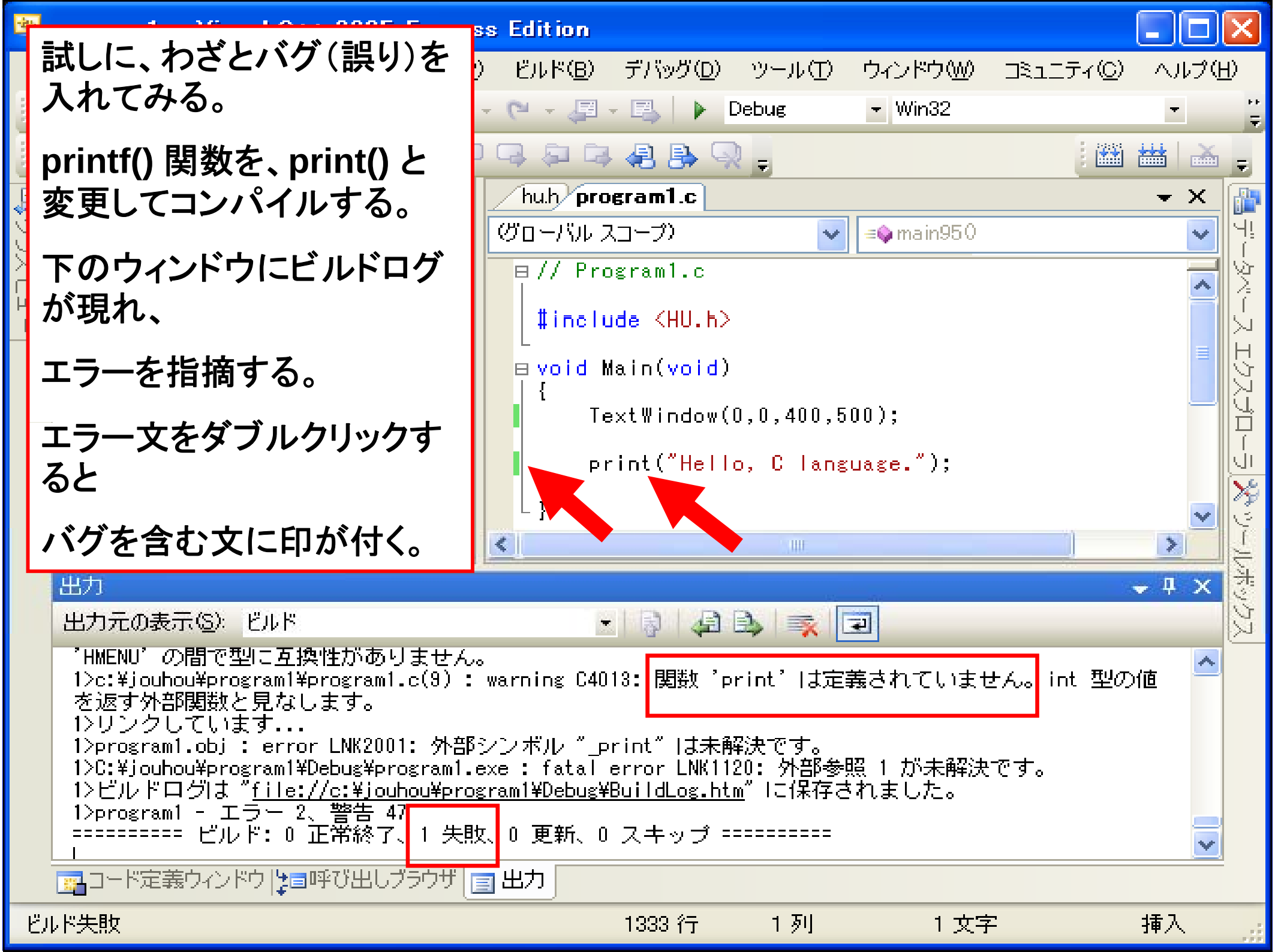
#include <HU.h> と記述されている部分を
#include “HU.h” と修正する。

プロジェクトフォルダ内のヘッダファイルの
読み込みは、

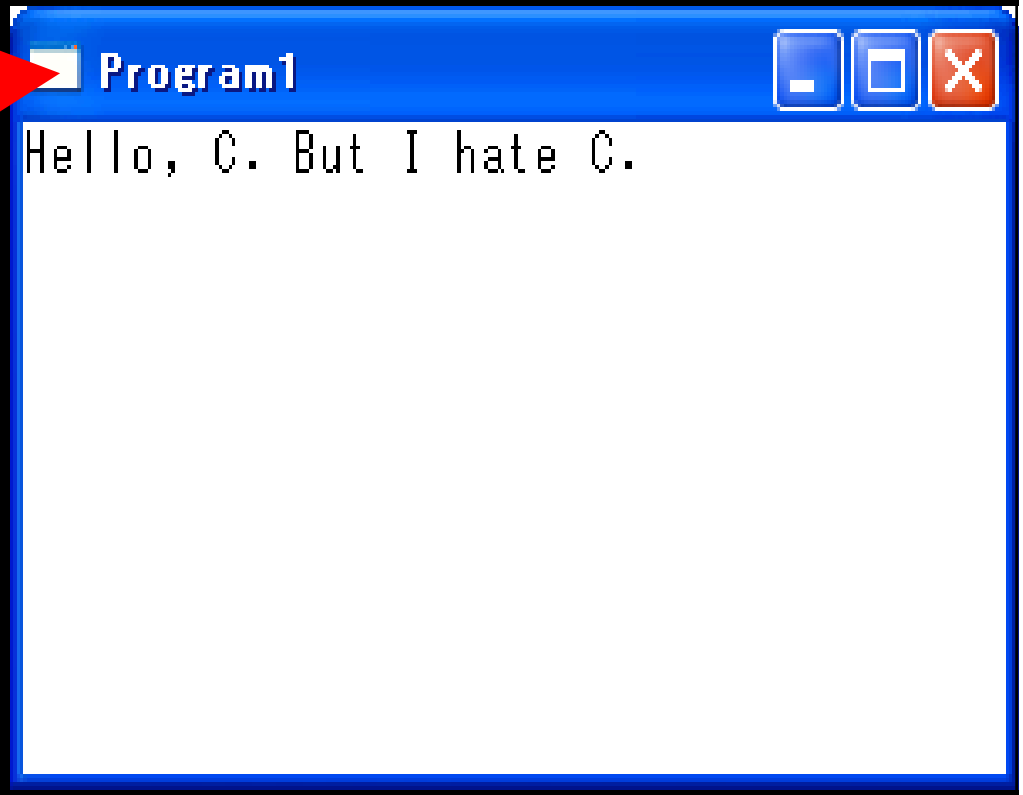
#include < > ではなく
#include “ ” と記述しないと読み込まない。

新たなプロジェクトフォルダを作るたびに、その中にヘッダファイルHU.hをコピーするのが面倒な場合は、C:¥Program Files ¥ Microsoft Visual Studio 9.0 ¥ VC ¥ include の フォルダ内に、HU.hをコピーすると、#include <HU.h> でOK。



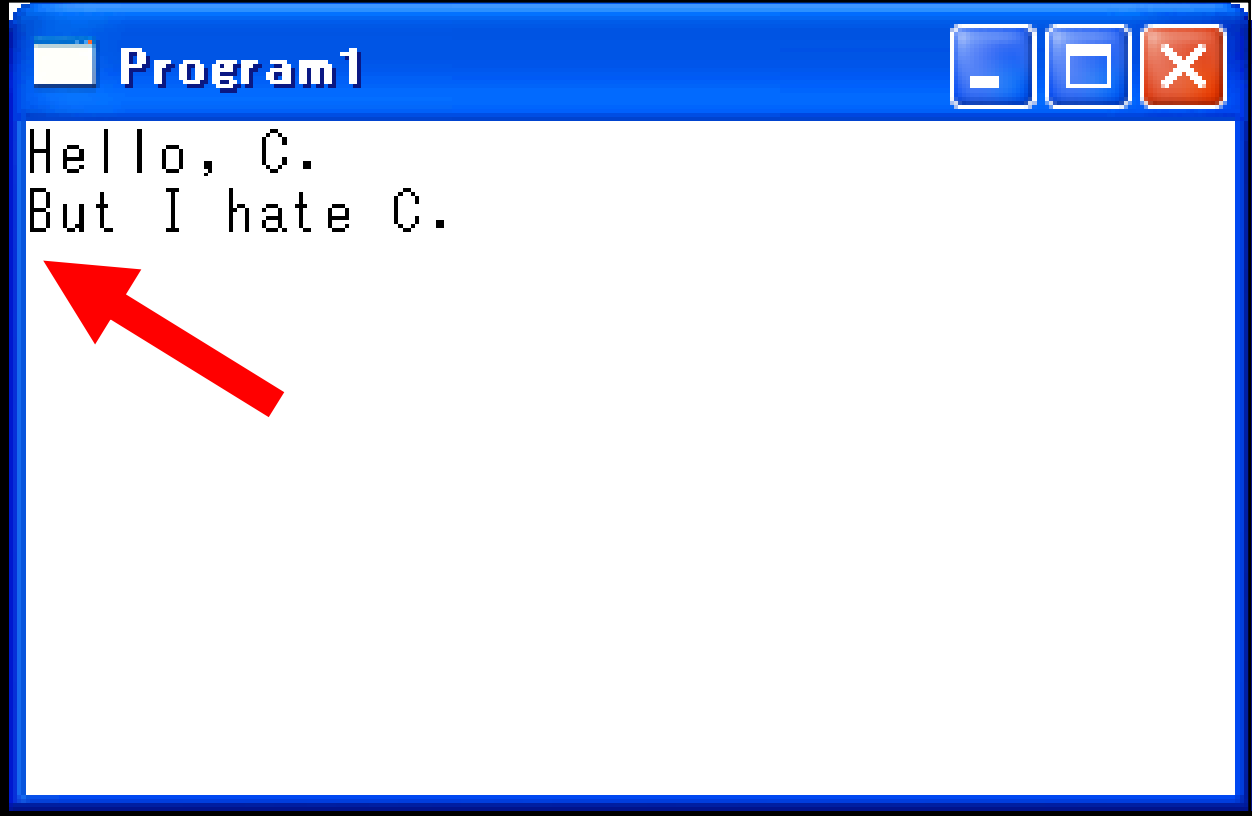

試しに、わざとバグ（誤り）を入れてみる。
printf() 関数を、print() と変更してコンパイルする。
下のウィンドウにビルドログが現れ、
エラーを指摘する。
エラー文をダブルクリックすると
バグを含む文に印が付く。
Debug
Win32
hu.h
program1.c
（グローバル スコープ）
main950
// Program1.c
#include <HU.h>
void Main(void)
{
TextWindow(0,0,400,500);
print("Hello, C language.");
}
出力
出力元の表示(S): ビルド
'HMENU' の間で型に互換性がありません。
1>c:¥jouhou¥program1¥program1.c(9) : warning C4013: 関数 'print' は定義されていません。int 型の値を返す外部関数と見なします。
1>リンクしています...
1>program1.obj : error LNK2001: 外部シンボル "_print" は未解決です。
1>C:¥jouhou¥program1¥Debug¥program1.exe : fatal error LNK1120: 外部参照 1 が未解決です。
1>ビルドログは "file://c:¥jouhou¥program1¥Debug¥BuildLog.htm" に保存されました。
1>program1 - エラー 2、警告 47
========== ビルド: 0 正常終了、1 失敗、0 更新、0 スキップ ==========
コード定義ウィンドウ
呼び出しブラウザ
出力
ビルド失敗
1333 行
1 列
1 文字
挿入

ウィンドウタイトル表示



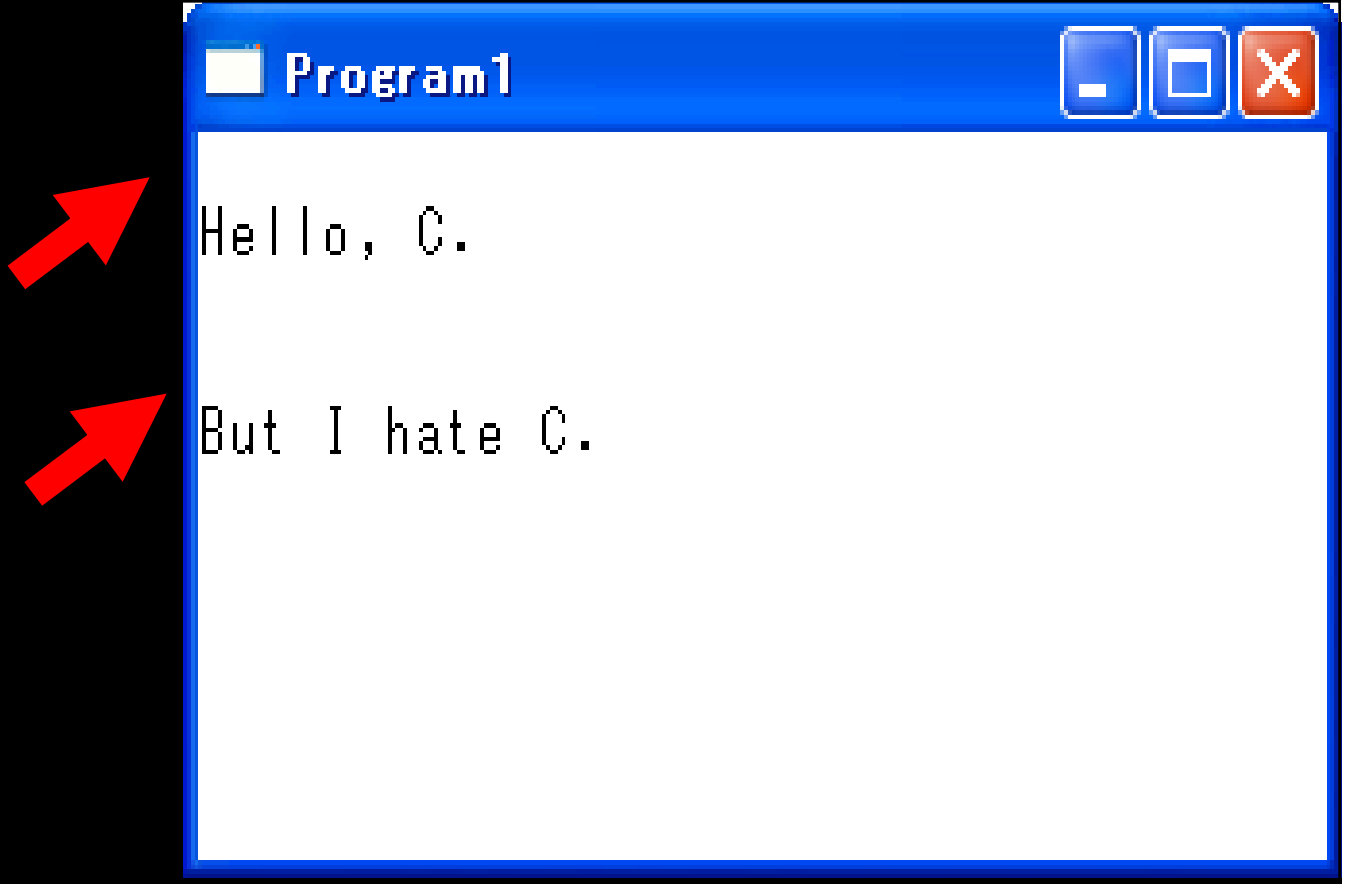

```
// Program1.c

#include <HU.h>

void Main(void)
{
        TextWindow(0,0,300,200);

        Title("Program1");

        printf("Hello, C. But I hate C.");
}
```

改行文字　¥n




```
// Program1.c

#include <HU.h>

void Main(void)
{
        TextWindow(0,0,300,200);

        Title("Program1");

        printf("Hello, C. ¥nBut I hate C.");
}
```

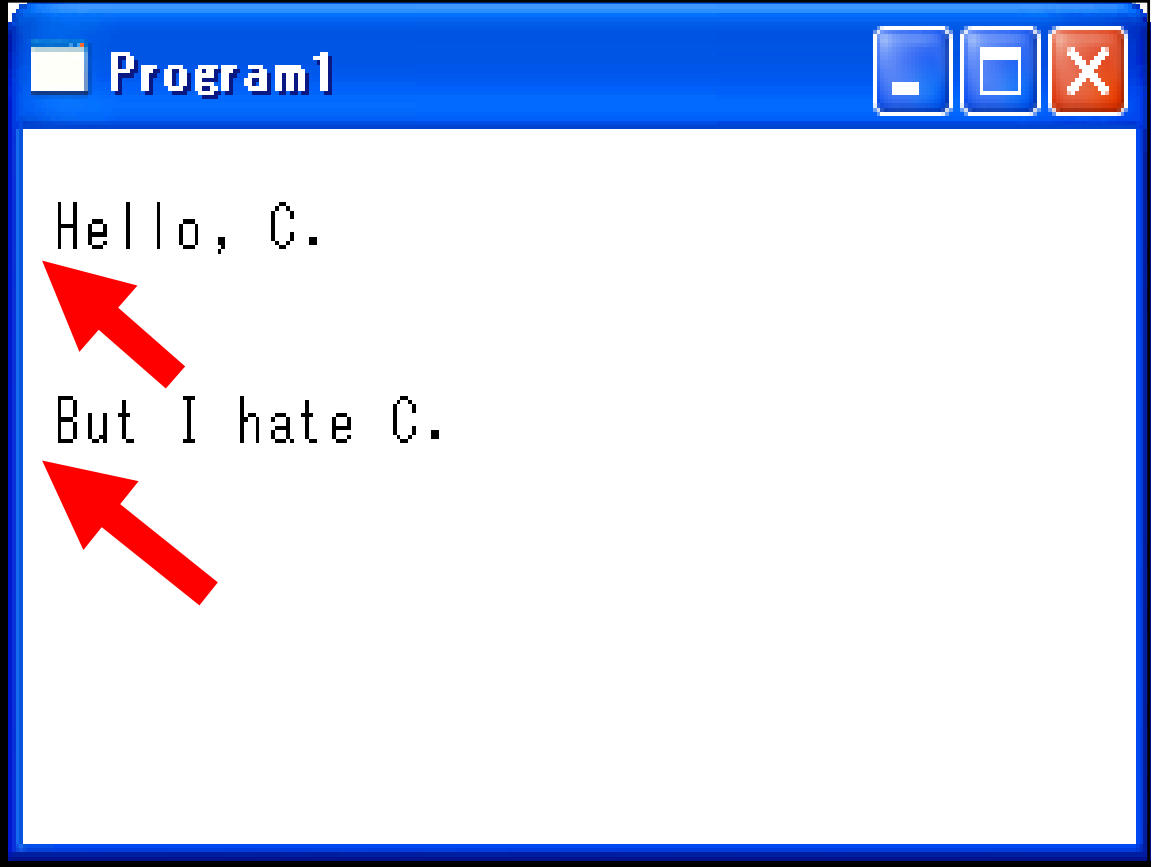
改行文字　¥n




```
// Program1.c

#include <HU.h>

void Main(void)
{
        TextWindow(0,0,300,200);

        Title("Program1");

        printf("¥nHello, C. ¥n¥n¥nBut I hate C.");
}
```

空白文字

```
// Program1.c

#include <HU.h>

void Main(void)
{
        TextWindow(0,0,300,200);

        Title(“Program1”);

        printf(“¥n  Hello, C. ¥n¥n¥n  But I hate C.");
}
```



Title(“ ”) や printf(“”) などの “ と ” に挟まれた
文字列を、文字列リテラルという。

文字列リテラルには日本語などの2バイト文字もOK.

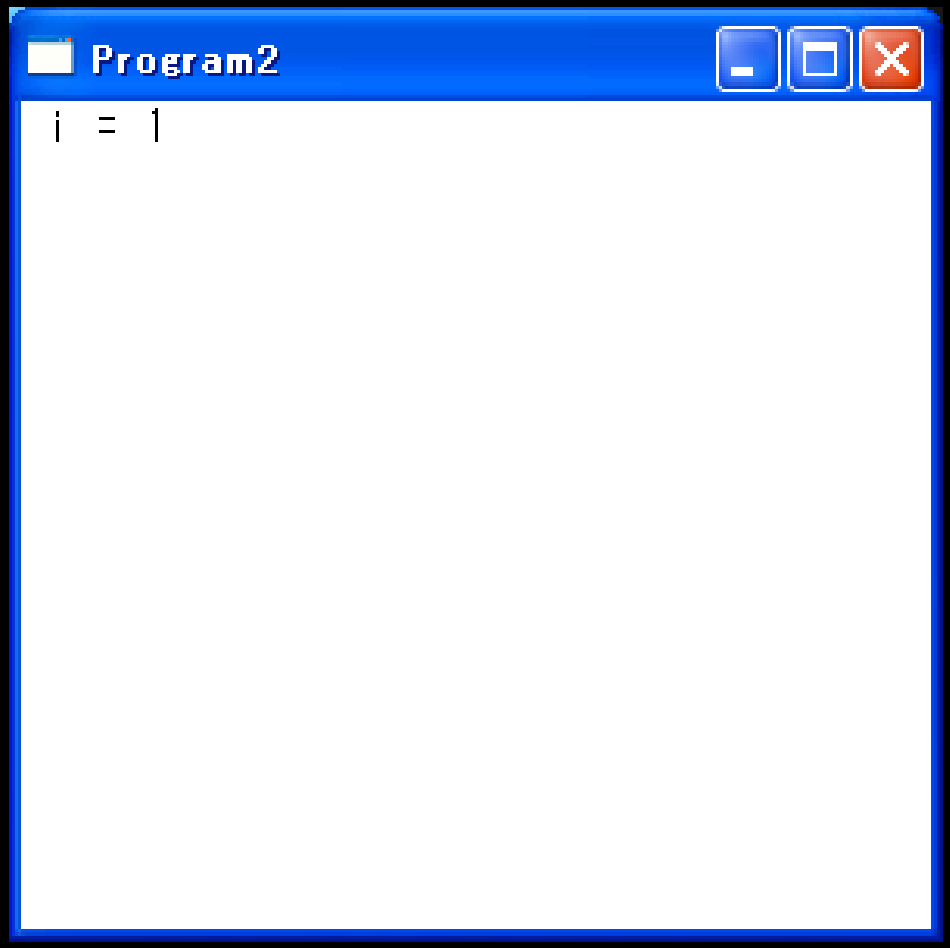
日本語を使ったときの注意点。 全角空白文字の恐怖。

日本語入力（全角文字入力）モードの状態で、
日本語を入れてはいけないところに空白文字を入れると、
普通の半角スペースとは異なる文字なので、
プログラムはエラーになります。

空白なので目で見ても間違いは見つかりません。
とても発見が困難なバグになります。
（バグ：bug = プログラム記述ミス）

以上の理由で、プログラム中は日本語を使わない方が無難。

```
// Program2.c     変数の宣言

#include <HU.h>

void Main(void)
{
        int i;

        TextWindow(0,0,300,300);

        Title("Program2");

        i=1;

        printf(" i = %d ", i);
}
```



int i ;　とは、

4バイト整数（1バイトは256なので256x256x256x256個の整数）を入れられる変数を使うことを宣言している。

通常の変数宣言文は、main関数内の最初に書く。

変数の値は、次の範囲内の整数です。

char 型（1バイト）は、-128 ～ 127 の範囲
（文字変数（半角英数文字））
short 型（2バイト）は、-32768 ～ 32767 の範囲
int 型（4バイト）は、-2147483648 ～ 2147483647 の範囲

float型は4バイトの実数型
double型は8バイトの実数型

i = 1 ;　とは、

整数変数 i に、数字　1　を代入している。

C言語では、= は、等号ではなく、代入を意味する。

(　C言語では、等号は、== と書く。不等号は、!= と書く。)

A = B ;　変数AにBの値を代入。(Aの値をBと同じにする。)

(A == B ;　AとBが等しい。)(A != B ;　AとBが異なる。)

i は aでも Bでも hensuu でもかまわない任意の変数名。

変数名は半角文字または半角文字列であれば何でもよいが、

普通、C言語では、変数は小文字、定数は大文字で宣言する。

printf(“ i = %d”,i); とは、

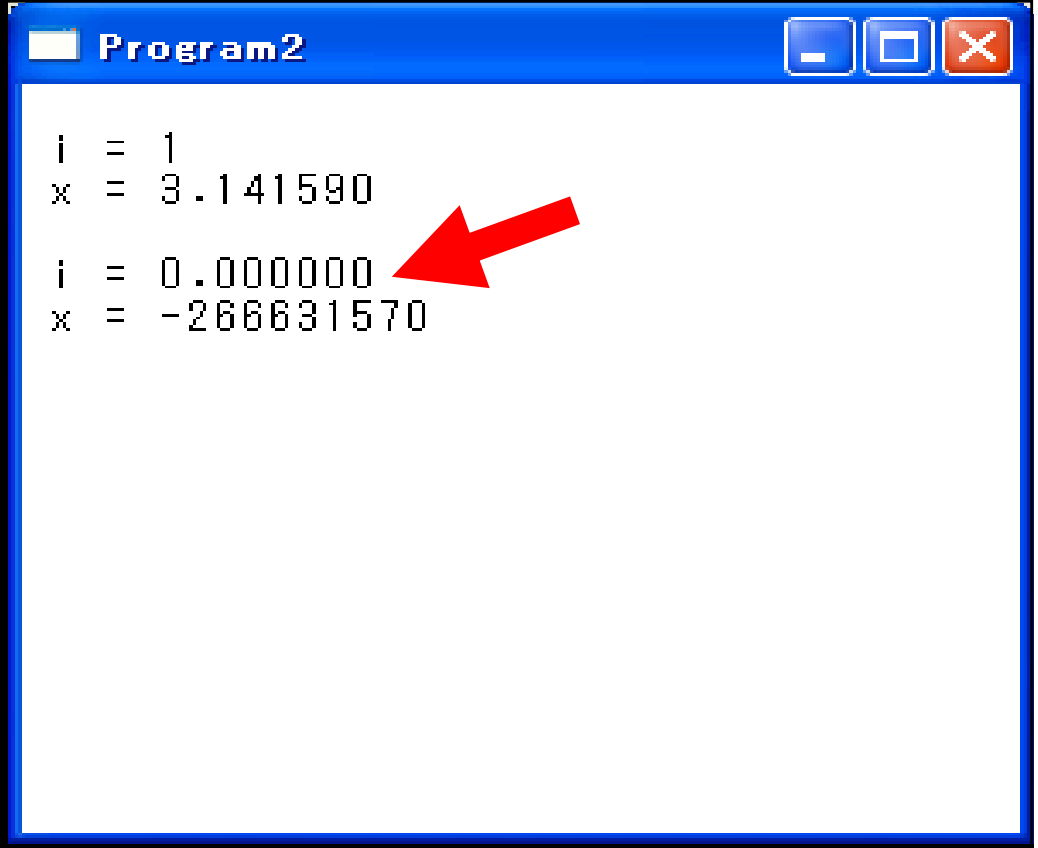
文字表示ウィンドウに、i = を表示し、
次に**整数を表示**することを示している。

printf（）関数は、整数を表示することを
%d と表現する。

文字変数を表示する場合は、**%c** と表現する。
実数を表示する場合は、**%lf** と表現する。
double x ; と宣言した場合は、 printf(“ x = %lf “, x) ;

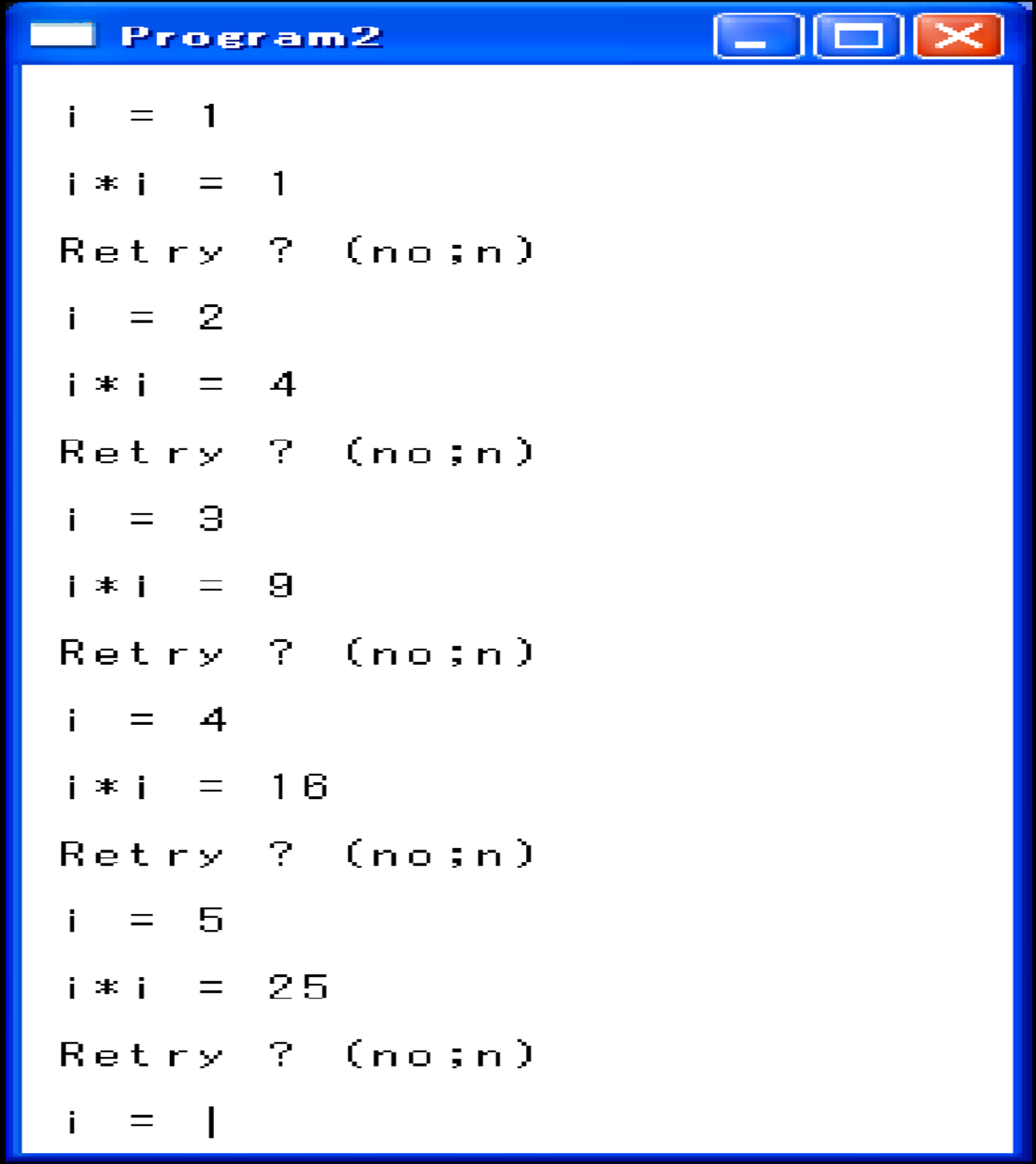
表示したい数字を持つ変数を、
” ”の次に ,（カンマ）を書いてから変数名を書いて、
カッコをとじる。

```
// Program2.c

#include <HU.h>

void Main(void)
{
        int             i ;
        double     x ;

        TextWindow(0,0,300,300);
        Title("Program2");

        i=1;  x = 3.14159;

        printf("¥n i = %d ", i);
        printf("¥n x = %lf", x);
        printf("¥n");
        printf("¥n i = %lf ", i);
        printf("¥n x = %d ", x);
}
```



誤った表示指定は
テキストウィンドウに
正しい数字を表示しない。
しかし文法エラーではない
のでプログラムは
動いてしまう。

```
// Program2.c                    整数の2乗を表示する

#include <HU.h>

void Main(void)
{
        char   yn;
        int    i;
    TextWindow(0,0,300,300);  Title("Program2");
  START:
        printf("¥n i = "); scanf("%d", &i);
        printf("¥n i*i = %d" , i*i);
        printf("¥n¥n Retry ? (no;n) "); scanf("%c",&yn);
        if( yn != 'n' ) goto START;
  exit(0);
}
```


Program2
i = 1
i*i = 1
Retry ? (no;n)
i = 2
i*i = 4
Retry ? (no;n)
i = 3
i*i = 9
Retry ? (no;n)
i = 4
i*i = 16
Retry ? (no;n)
i = 5
i*i = 25
Retry ? (no;n)
i =

printf(“¥n i = ”); scanf(“%d”, &i) ;

テキストウィンドウに　i = と表示し、その右側に
キーボードで入力した整数値を、変数 i に入れる。

scanf () 関数は、文字表示ウィンドウから、変数を
入力できる関数。
“ %d ” は、printf () 関数と同じく、整数を扱うこと
を示している。
実数を扱う場合は、printf () 関数と同じく、
“%lf” と書く。
文字変数を扱う場合は、printf () 関数と同じく、
“%c” と書く。

数字を入力したい変数の型を、” ” の中に記述して、
次に ,(カンマ)を書いてから**変数のポインタ**を書いて、
カッコをとじる。

変数のポインタとは、

変数が、コンピュータのメモリ内に置かれている
アドレス（先頭番地）を示す。

scanf()関数は、キーボードから入力された
変数の型や、メモリ内のどこに変数がいるかを
知らないので、それを明記する必要がある。

変数名の頭に＆（アンパサンド）を書くと、変数の
メモリ内に置かれているアドレス（番地）を示し、
ポインタと呼ばれる。

printf("¥n¥n Retry ? (no;n) "); scanf("%c",&yn);

テキストウィンドウで Retry ? (no;n) と表示された右側にキーボードで入力された**文字**を、**文字変数** yn に入れる。

if(yn != ‘n’) goto START;

文字変数 yn が 文字 n と違うとき、ラベルSTART: の所にプログラム実行部位を変更する。人が n と認識する**文字**は、プログラム中では ‘n’ と記述する。

IF文　if(A) B;

式Aが正しければ、プログラム文Bを実行する。
式Aが正しくなければ、プログラム文Bを無視する。

実行文を複数記述したい場合は **if(A) { B; C; D; }** と中括弧 { } で囲む。

ラベル名: ～ goto ラベル名;

ラベル(名札)はプログラム文ではないので、
終端はセミコロンではなく、コロン: を記述する。

goto ラベル名; はプログラム文なので終端は ;

ラベル名は、適当な文字列を記述できるが、
大文字で目立つように書くのが普通。
インデントを下げて見やすいようにする。

goto 文 が指すラベルまで プログラム文実行の順序を
間単に変更でき便利だが、計算ループ中などに使うと
予期せぬ計算ミスの原因になりやすく、十分に注意して
使用する必要がある。

exit(0);

プログラムを終了させるプログラム文。

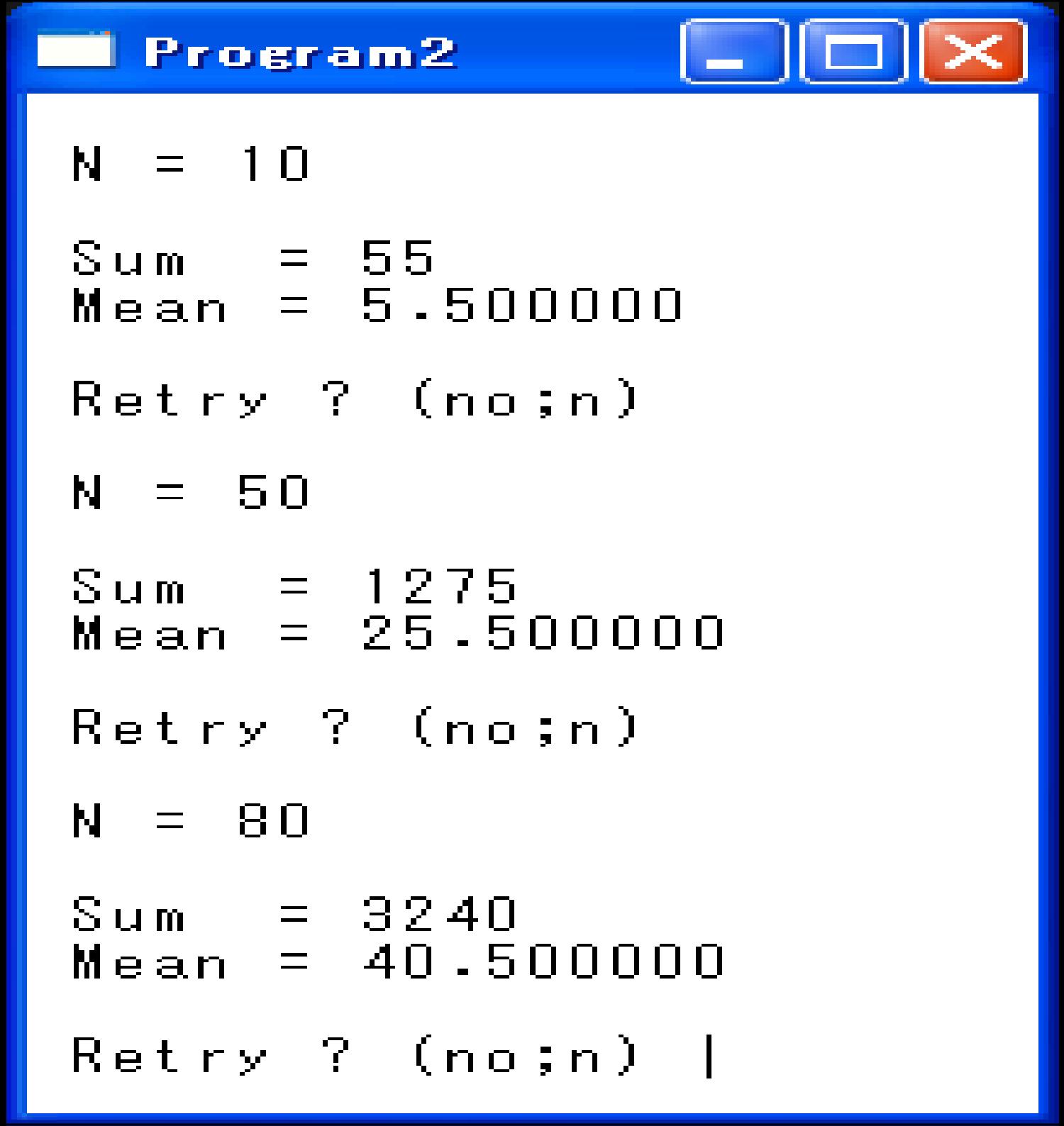
exit()は、関数が出力する値を設定するものだが、
main()関数の出力は定義されていないので
（void と宣言済み。 void Main(void) ）、
便宜上 0 を出力すると記述している。
カッコ内の数字が関数の出力値。
0でなくても構わないが、main関数の終了は
exit(0); で記述するのが普通。

Basic言語で使う End と同じはたらき。

整数1からNまでの
総和と平均を計算する

```
// Program2.c
#include <HU.h>
void Main(void)
{
        char        yn;
        int         i, N, sum;
        double      mean;

        TextWindow(0,0,300,300);
        Title("Program2");
START:
        printf("¥n N = "); scanf("%d", &N);
```

```
    sum=0;
    for( i=1; i<=N; i++ ){
        sum += i;
    }
    mean = (double)sum / (double)N;
    printf("¥n Sum  = %d" , sum);
    printf("¥n Mean = %lf", mean);

    printf("¥n¥n Retry ? (no;n) ");  scanf("%c",&yn);

    if( yn != 'n' ) goto START;

  exit(0);
}
```


Program2
N = 10
Sum = 55
Mean = 5.500000
Retry ? (no;n)
N = 50
Sum = 1275
Mean = 25.500000
Retry ? (no;n)
N = 80
Sum = 3240
Mean = 40.500000
Retry ? (no;n)

```
int          i, N, sum;
```

同じ型の変数は、コンマ，で区切って
１行でまとめて宣言できる。

```
for( i=1; i<=N; i++ ) {          }
```

変数 i が 1 から N まで 1 ずつ増加しながら
中括弧 { } 内のプログラム文を N 回実行する。

For 文　画像を扱うプログラムでは最も多く使われる構文

```
For(初期条件; 実行条件; 更新条件 ){
    プログラム文;
}
```

まず初期条件の状態でプログラム文が実行される。

次に更新条件の式が実行され、

2回目のプログラム文実行が行われる。

実行条件が満たされている間、実行が繰返される。

```
sum=0;
for( i=1; i<=N; i++ ){
    sum += i;
}
```

整数変数sumに １＋２＋・・・＋N が入る。

C言語の特殊な計算式表記法。

i++ は i = i+1 と同じ。i の値が1づつ増加する。

sum += i は sum = sum + i と 同じ。

（普通の表記法でも構わないが、慣れると便利。）

変数の型変換（キャスト cast）

```
mean = (double)sum / (double)N;
```

総和sum と データ数 N は、ともに整数変数。
平均を入れる実数変数meanには整数は入らない。
総和／データ数　の値を代入するが、
整数を整数で割った値は整数になり、誤差が多い。

そこで、**変数の型変換を行う。（キャスト cast）**

（double）を整数変数の前に記述すると実数変数になる。

```
// Program2.c          整数1からNまでの平均、標準偏差を計算
#include <HU.h>

void Main(void)
{
        char            yn;
        int             i, N, sum;
        double          mean, v, sd;

        TextWindow(0,0,300,300); Title("Program2");

  START:

        printf("¥n N = "); scanf("%d", &N);


        sum=0; v =0.0;

        for(i=1; i<=N; i++ ){   sum += i; }

        mean = (double)sum / (double)N;
```

```
        for(i=1; i<=N; i++ ){

                v += (mean - (double)i) * (mean - (double)i);
        }

        v /= (double)N;          //  v = v/(double)N

        sd = sqrt(v);            // sqrt( )  平方根を求める関数

        printf("¥n Sum       = %d ", sum );     // 総和
        printf("¥n Mean      = %lf", mean);     // 平均
        printf("¥n Variation = %lf", v    );     // 分散
        printf("¥n Deviation = %lf", sd   );     // 標準偏差

        printf("¥n¥n Retry ? (no;n) "); scanf("%c",&yn);
        if(yn!='n') goto START;

    exit(0);
}
```

sum = 0 ; v = 0.0 ;

sum, v ともに 総和を求めるために用意した変数なので、
はじめは 数値 0 を代入しておく必要がある。

変数を宣言したときに 初期値が0 とは保証されていない。

sum は int で宣言された整数変数。

v は double で宣言された（倍精度）実数変数。

数字 0 は、数値 0 の 整数表記。
数字 0 は整数なので　sum = 0 ;　は正しいプログラム文。

実数変数 v には 整数を代入できないので、
数値 0 の実数表記 0.0 (または 0.) を記述する。

v = 0.0 ; または v = 0. ; と記述すると、
変数 v に実数値の 0 が代入される。

```
for( i=1; i<=N; i++ ){
   v += ( mean - (double)i ) * (mean - (double)i );
}
```

変数 v に、平均との差の2乗を足していく forループ文。

mean は 実数変数なので、整数変数 i を直接引くことはできない。

（double）を付けて、i を実数にキャストしてから引く。

`v += ( mean - (double)i ) * (mean - (double)i );` は、
`v = v + ( mean - (double)i ) * (mean - (double)i );` と同じ。

変数 v に、差の２乗が次々と加算され、
v の値が増加していくことを理解して下さい。

記号 = は、等号ではなく、代入を示すことを理解する。

課題 2

0 からN までの、0.1刻みの実数 0.0, 0.1, 0.2, … N の総和、平均、分散、標準偏差を求めるプログラム program3.c を書いて、メールに添付してホームページの課題提出ボタンから今週中にメールを送って下さい。
または　chtgkato@.med.hokudai.ac.jp　へ。

メールの件名には、学生番号、氏名、課題番号を明記。

可能ならば実行ウィンドウを Winshot などのフリーウェア等を使ってjpgファイル（300KB以下）にして添付。

さらに可能ならば、自分で好きなようにプログラムを改造して送って下さい。