- BorlandC++5.5、TurboC++2006で動作させる場合
- GCC（MinGW Ver3.4.5）で動作させる場合

Ⅰ BorlandC++，TurboC++で実行する上での注意事項

この附録で扱っている処理系は以下のものである．

- BorlandC++（アプリケーション種類：コマンドライン）
- TurboC++ （アプリケーション種類：コンソールアプリケーション，VCL）

今回の改訂にあたって本書のプログラムはBorlandC++Ver5.5，TurboC++2006で動作確認を行った．

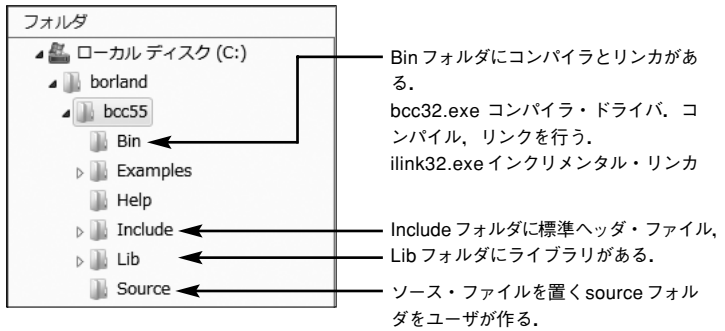
Ⅱ コマンドライン，コンソールアプリケーション

1．BorlandC++コマンドライン

対象プログラム： Dr17，8章（グラフィックス）以外のプログラム

本書のプログラムはそのまま動作する．

①BorlandC++ コマンドライン版（フリー・ソフト）をインストールすると以下のようなフォルダ構成になる．



②以下のようなバッチ・ファイルを作っておくと便利である．

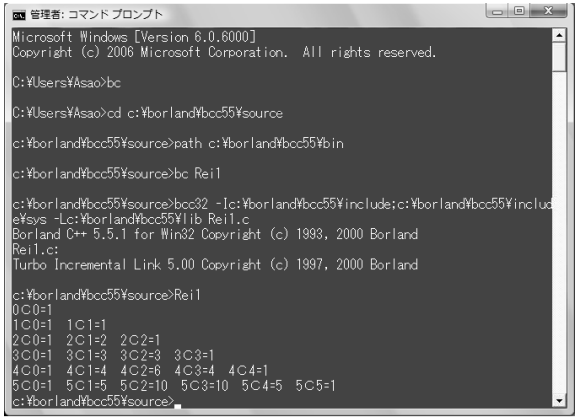
- 環境設定バッチファイル（bc.bat：コマンドプロンプトのデフォルトフォルダに置く）

```
cd c:¥borland¥bcc55¥source
path c:¥borland¥bcc55¥bin
```

- コンパイル・リンク実行バッチファイル（bc.bat：borland¥bcc55¥source フォルダに置く）

  -Iオプションでインクルード・ファイルのフォルダ，-Lオプションでライブラリのフォルダを指定する．

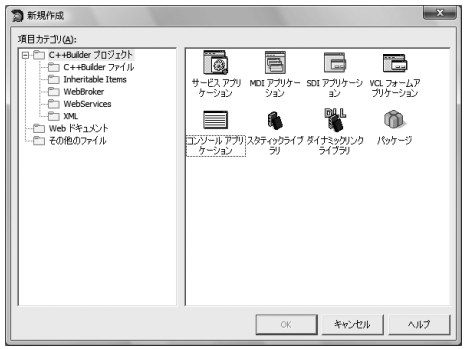
```
bcc32 -Ic:¥borland¥bcc55¥include;c:¥borland¥bcc55¥include¥
sys -Lc:¥borland¥bcc55¥lib %1.c
%1
```



2．TurboC++ コンソールアプリケーション

対象プログラム：Dr17，8章（グラフィックス）以外のプログラム

①［ファイル(F)］－［新規作成(N)］－［その他(O)...］－［コンソールアプリケーション］を選択する．



②以下のスケルトンが生成される．

```
#include <stdio.h>
#pragma hdrstop
#pragma argsused
int main(int argc, char* argv[])
{
    return 0;
}
```


③このスケルトンの代わりに本書のプログラム（メイン関数にgetchar();を追加したもの）をそのまま置き換えるかスケルトンのmainを残し，その中にコードを書く．

④［実行］メニューか［実行］ボタンでビルドする．

TurboC++ コンソールアプリケーションで開いたDOS窓がプログラム終了で自動的に閉じてしまうので，メイン関数の終りに`getchar();`を置き Enter キーの入力を待ってDOS窓が閉じるようにする．

【注意事項】

- scanf関数を使って入力しているものは`getchar();getchar();`のように2つ置く．または`rewind(stdin);getchar();`

  該当プログラム

  Rei6，Dr6，Rei9，Dr9，Rei20，Dr20_1，Dr20_2，Rei21，Dr21，Dr26_4，Dr26_5，Rei29，Dr29，Rei42，Dr42，Rei55，Dr55

- scanf関数のループで Ctrl ＋ Z で終了するタイプのものは`getchar();`を置かなくてもよい．

  該当プログラム

  Rei7，Dr7_3，Rei25，Dr25，Rei32，Dr32，Rei33，Dr33，Dr35_2，Rei40，Dr40，Rei41，Dr41，Dr44，Rei50，Dr50_1，Dr50_2，Rei65，Dr65

Ⅲ TurboC++ VCLフォームアプリケーション

対象プログラム： Dr17，8章（グラフィックス）

TurboC++用にグラフィックスライブラリを用意した．Includeフォルダ内にこのライブラリヘッダーファイルを置くか，カレントフォルダに置く．名前は`glib.h`に変更するか，`turboglib.h`を使う．

このライブラリはフォームが表示されというイベントによりフォームに対して描画処理を行う．イベントを受けて動作する関数をイベントハンドラとかメッセージハンドラと呼んでいるが，この附録ではイベントハンドラという言葉を用いた．

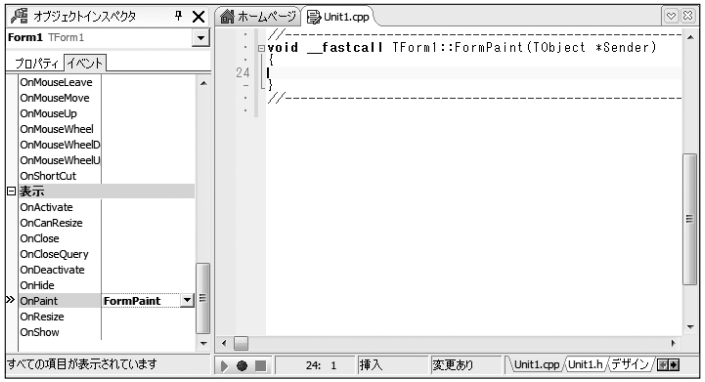
プロジェクトの作り方

①［ファイル(F)］－［新規作成(N)］－［VCL フォームアプリケーション-C++Builder(V)］を選ぶ．

②［表示(V)］メニューから［オブジェクトインスペクタ］を開く．

③［オブジェクトインスペクタ］の［イベント］タブを開き，「OnPaint」の右側をクリックすると，FormPaint イベントハンドラのスケルトンが作成される．

④FormPaint イベントハンドラの前に#include "turboglib.h"を置く．



⑤メイン関数内のコードをFormPaint イベントハンドラ内に記述する．

プログラムの記述例

```
#include "turboglib.h"          ←――― 追加
void __fastcall TForm1::FormPaint(TObject *Sender)
{
    // この位置にコードを記述
                                ←――― main関数のコードを記述
}
```

●turboglib.h（TurboC++ VCL フォームアプリケーション版）

```
/*
 * ------------------------------------------
 *      基本グラフィックスライブラリ          *
 *  TurboC++ VCLフォームアプリケーション版    *
 * ------------------------------------------
 */

#include <math.h>

TCanvas* gCanvas;            /* グラフィックオブジェクト */

double WX1,WY1,WX2,WY2,      /* ワールド座標 */
       VX1,VY1,VX2,VY2,      /* ビュー座標   */
       FACTX,FACTY,          /* スケール     */
       ANGLE,                /* 現在角       */
       LPX,LPY;              /* 現在位置     */

void window(double x1,double y1,double x2,double y2)
{
    WX1=x1; WY1=y1; WX2=x2; WY2=y2;
    FACTX=(VX2-VX1)/(WX2-WX1);
    FACTY=(VY2-VY1)/(WY2-WY1);
}
void view(int x1,int y1,int x2,int y2)
{
    VX1=(double)x1; VY1=(double)y1; VX2=(double)x2; VY2=(double)
y2;
    FACTX=(VX2-VX1)/(WX2-WX1);
    FACTY=(VY2-VY1)/(WY2-WY1);
}
void Ginit(TCanvas* canvas)
{
    gCanvas=canvas;
    gCanvas->Pen->Color=clBlue; // 青のペン
    LPX=0;LPY=0;ANGLE=0;
    window(0,0,639,399);
    view(0,0,639,399);
}
void Cls(int x,int y)
```

```
{
    gCanvas->Brush->Style=bsSolid;
    gCanvas->FillRect(Rect(0,0,x,y));
}
void line(double x1,double y1,double x2,double y2)
{
    int px1,py1,px2,py2;
    px1=(int)((x1-WX1)*FACTX+VX1);
    py1=(int)((WY2-y1)*FACTY+VY1);
    px2=(int)((x2-WX1)*FACTX+VX1);
    py2=(int)((WY2-y2)*FACTY+VY1);
    gCanvas->MoveTo(px1,py1);
    gCanvas->LineTo(px2,py2);
    LPX=x2;LPY=y2;
}
void pset(double x,double y)
{
    int px,py;
    line(x,y,x+1,y);
    LPX=x;LPY=y;
}
void move(double l)
{
    double x,y,rd=3.1415927/180;
    x=l*cos(rd*ANGLE);y=l*sin(rd*ANGLE);
    line(LPX,LPY,LPX+x,LPY+y);
}
void moveto(double x,double y)
{
    line(LPX,LPY,x,y);
}
void setpoint(double x,double y)
{
    LPX=x;LPY=y;
}

#define setangle(a) ANGLE=(double)(a)
#define turn(a) ANGLE=fmod(ANGLE+(a),360.0)

#define ginit() Ginit(Canvas)
#define cls() Cls(ClientWidth,ClientHeight)
```

【注意事項】

- ginitは，Canvasをマクロ展開しているので必ずFormPaintイベントハンドラの中に置かなければならない．

- 描画色は青に設定してあるが，変更したい場合は以下の`clBlue`の値を変える．

```
gCanvas->Pen->Color=clBlue;
```

- TurboC++でクリップ領域の指定を行う方法がわからなかったのでview関数にこの処理を含めていない．

□フォーム画面へのテキストの出力（Dr17）

Dr17のようにフォームへのグラフィックスとテキストを出力する場合は，グラフィックスライブラリでグラフィック描画を行いprintf関数の出力は以下のように行う．改行幅はフォントに応じて適当に設定する．詳しくは「Ⅳ　GUI環境を使ったプログラムへの移植その1」を参照．

```
/* 係数の表示 */
char buf[80];
for (k=0;k<=M;k++){
    sprintf(buf,"a%d=%f",k,a[k][M+1]);
    Canvas->TextOut(0,k*20,buf);
}
```

Ⅳ TurboC++ GUI環境を使ったプログラムへの移植その1

TurboC++のコンソールアプリケーション用コードをGUI環境に移植するにあたり，以下の2点を考慮したライブラリを作成する．

1．コンソール出力を行うprintf関数，putchar関数は，そのままソースコード中に置き，ライブラリでマクロ機能などを使ってピクチャーボックス（ピクチャーコントロール）への出力を行えるようにする．たとえば

```
printf("%3d%5.1f¥n",i,x);
```

を使ってピクチャーボックスへの出力を行う基本的な仕組みは以下の通り．

- printfをPrintfにマクロ置換する．
- Printf関数では可変引数マクロ等を使いprintfの書式付出力を文字列に展開，pDCデバイスコンテキストのTextOut関数（DrawString関数）を使ってこの展開した文字列をTLPX,TLPY位置に描画する．変数TLPX,TLPYがテキストの描画現在位置を示す．
- printf関数の書式制御文字列中の最後に`¥n`がある場合または単独の`printf("¥n")`に対して改行動作が行える．
- `putchar(a)`は`printf("%c",a)`にマクロ置換することでprintf関数の処理に置き換える．
- フォントの高さで改行し，フォント幅で文字位置を横に進める．
- 80文字出力したら改行する（フォント幅の設定の問題から例題，練習問題によっては正確に80文字で改行しない場合がある）．

2．コンソール入力を行うscanf関数はテキストボックスから入力を行うプログラムに置き換える．その際テキストボックスで扱う文字列型（AnsiString）を各型に変換する以下の関数をライブラリに含める．

`stoc(s,c)`
テキストボックスで扱う文字列型`s`をchar配列`c[]`に変換．ASCII文字のみサポート．

`stoi(s)`

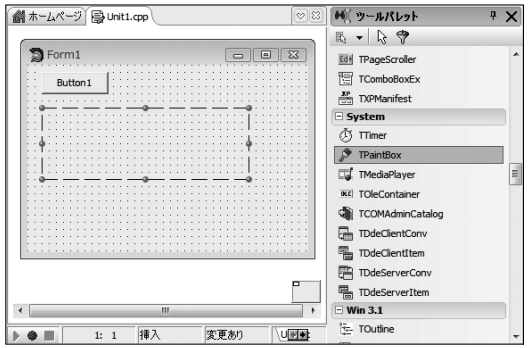
テキストボックスで扱う文字列型`s`を`int`値に変換して返す．

`stod(s)`
テキストボックスで扱う文字列型`s`を`double`値に変換して返す．

プロジェクトの作り方

①［ファイル(F)］－［新規作成(N)］－［VCLフォームアプリケーション-C++Builder(V)］を選ぶ．

②ツールパレットから，フォームにボタンとペイントボックスを配置する．
［表示(V)］－［ツールパレット］，［表示(V)］－［オブジェクトインスペクタ］で各ウィンドウを表示することができる．ペイントボックスのボーダースタイルは指定できない．



③配置したボタンをダブルクリックし，Button1Clickイベントハンドラのスケルトンを生成する．

④本書に記載のプログラムを以下の要領で入力する．

- Button1Clickイベントハンドラの前に，以下を置く．
  コメント
  `#include "turboform.h"`
  main関数外の変数宣言，`#define`指令
  ユーザ関数定義
  ※turboform.hの中でstdio.h，string.h，stdarg.h，math.hをインクルードするので重複するものは指定しないでよい．

- 関数定義をButton1イベントハンドラの前に置いたため関数プロトタイプ宣言は削除する．

- main関数内のコードはButton1イベントハンドラ内に記述する．

- フォームへの出力を行うための初期設定として，`tinit();cls();`を変数宣言の後でprintf関数の使用前に置く．
  ※後始末処理を行うtfin();は置かなくてよい．

プログラムの記述例 Rei9（テキストボックスからの入力を伴う部分）

```
//---------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Rei9.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------
#include "turboform.h"  <-- 追加

#define f(x) (sqrt(4-(x)*(x)))  <-- 変数宣言、#define指令
                                <-- ユーザ関数があればここに記述
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    int k;
    double a,b,n,h,x,s,sum;
    a=stod(Edit1->Text);   <-- 追加
    b=stod(Edit2->Text);
    n=50;
    h=(b-a)/n;
    x=a; s=0;
    for (k=1;k<=n-1;k++){
        x=x+h;
        s=s+f(x);
    }
    sum=h*((f(a)+f(b))/2+s);
    tinit();cls();          <-- 追加
    printf("   /%f¥n",b);
    printf("   |  sqrt(4-x*x) =%f¥n",sum);
    printf("   /%f¥n",a);
```

（main関数のコードを記述）

```
}
//---------------------------------------------------------------------
```

●turboform.h（TurboC++ VCLフォームアプリケーション版）

```
/*
 * ----------------------------------------
 *  TurboC++ VCLフォームアプリケーション版    *
 *   テキスト＆グラフィックス・ライブラリ     *
 * ----------------------------------------
 */

#include <stdio.h>
#include <string.h>
#include <stdarg.h>
#include <math.h>

TCanvas *gCanvas;   /* グラフィックスオブジェクト */

double WX1,WY1,WX2,WY2,      /* ワールド座標 */
       VX1,VY1,VX2,VY2,      /* ビュー座標   */
       FACTX,FACTY,          /* スケール     */
       ANGLE,                /* 現在角       */
       LPX,LPY;              /* 現在位置     */

int TLPX,TLPY;     /* テキスト描画現在位置 */
float fonth,fontw; /* フォントの高さと幅   */

/* ------------------------- */
/*  グラフィックスライブラリ    */
/* ------------------------- */

void window(double x1,double y1,double x2,double y2)
{
    WX1=x1; WY1=y1; WX2=x2; WY2=y2;
    FACTX=(VX2-VX1)/(WX2-WX1);
    FACTY=(VY2-VY1)/(WY2-WY1);
}
void view(int x1,int y1,int x2,int y2)
{
    VX1=(double)x1; VY1=(double)y1; VX2=(double)x2; VY2=(double)
y2;
    FACTX=(VX2-VX1)/(WX2-WX1);
    FACTY=(VY2-VY1)/(WY2-WY1);
}
void Ginit(TCanvas* canvas)
{
    gCanvas=canvas;
    gCanvas->Pen->Color=clBlue; // 青のペン
    LPX=0;LPY=0;ANGLE=0;
    window(0,0,639,399);
    view(0,0,639,399);
}
```

```
void Cls(int x,int y)
{
    gCanvas->Brush->Style=bsSolid;
    gCanvas->FillRect(Rect(0,0,x,y));
    TLPX=TLPY=0;
    ANGLE=LPX=LPY=0;
}
void line(double x1,double y1,double x2,double y2)
{
    int px1,py1,px2,py2;
    px1=(int)((x1-WX1)*FACTX+VX1);
    py1=(int)((WY2-y1)*FACTY+VY1);
    px2=(int)((x2-WX1)*FACTX+VX1);
    py2=(int)((WY2-y2)*FACTY+VY1);
    gCanvas->MoveTo(px1,py1);
    gCanvas->LineTo(px2,py2);
    LPX=x2;LPY=y2;
}
void pset(double x,double y)
{
    int px,py;
    line(x,y,x+1,y);
    LPX=x;LPY=y;
}
void move(double l)
{
    double x,y,rd=3.1415927/180;
    x=l*cos(rd*ANGLE);y=l*sin(rd*ANGLE);
    line(LPX,LPY,LPX+x,LPY+y);
}
void moveto(double x,double y)
{
    line(LPX,LPY,x,y);
}
void setpoint(double x,double y)
{
    LPX=x;LPY=y;
}

#define setangle(a) ANGLE=(double)(a)
#define turn(a) ANGLE=fmod(ANGLE+(a),360.0)

// ユーザ関数でPaintBox1を参照するにはForm1->が必要
#define ginit() Ginit(Form1->PaintBox1->Canvas)
#define cls() Cls(Form1->PaintBox1->ClientWidth,Form1->PaintBox1
->ClientHeight)


/* ------------------------ */
/*  テキスト出力ライブラリ    */
/* ------------------------ */


void Tinit(TCanvas* canvas)
```



```
{
    gCanvas=canvas;
    gCanvas->Font->Size=10;
    gCanvas->Font->Name="ＭＳ ゴシック";
    fonth=gCanvas->Font->Height;
    if (fonth<0) {
        fonth=-fonth;
    }
    fontw=fonth/1.8;
    TLPX=0;
    TLPY=0;
}
void Printf(char *format,...)
{
    char buf[80];
    int n;
    va_list ap;
    va_start(ap,format);
    n=vsprintf(buf,format,ap);
    va_end(ap);
    if (TLPX>=80*fontw){
        TLPX=0;
        TLPY+=fonth;
    }
    if (buf[strlen(buf)-1]=='¥n') {
        buf[strlen(buf)-1]='¥0';
        gCanvas->TextOutA(TLPX,TLPY,buf);
        TLPY+=fonth;
        TLPX=0;
    }
    else {
        gCanvas->TextOutA(TLPX,TLPY,buf);
        TLPX+=n*fontw;
    }
}
void PrintfL(void)
{
    TLPY+=fonth;
    TLPX=0;
}
#define tinit() Tinit(Form1->PaintBox1->Canvas)
#define printf Printf
#define putchar(x) Printf("%c",x)

void stoc(AnsiString s,char *p)
{
    int i,n;
    wchar_t buf[80];
    n=s.Length();
    s.WideChar(buf,n);
    for (i=0;i<n;i++) {
        p[i]=(char)buf[i];
    }
    p[i]='¥0';
```

```
}

#define stoi(s) s.ToInt()
#define stod(s) s.ToDouble()
```

【注意事項】

- グラフィックスライブラリは`turboglib.h`をピクチャーボックスへの描画に変更した．`cls`の中に現在位置（グラフィックス，テキスト共），現在角を初期設定する処理を置いた．

- TurboC++でクリップ領域の指定を行う方法がわからなかったのでview関数にこの処理を含めていない．

- TLPXの初期値は0とした．

- 文字幅をフォントの高さ`/1.8`と仮定したがフォントの種類によって変更が必要．

- MFC版のような後始末処理を行う`tfin();`は置かなくてよい．

- putcharがマクロの二重定義となる場合は以下のようにする．

```
#undef putchar
#define putchar(x) Printf("%c",x)
```

移植上の細部な注意点

- 関数プロトタイプ宣言を置かない場合に，関数の定義順序を変える．

  該当プログラム

  Rei24，Dr24，Dr32，Rei44，Dr44_1，Dr44_2，Rei45，Dr45_1，Dr45_2，Rei46，Dr46

- main関数の中に書かかれている処理が複数のイベントハンドラに分かれる場合に各イベントハンドラで共通に使う変数は関数の外で宣言する．たとえばRei34の`struct tfield *head，*p;`の`head`は外で宣言し，`p`は個々のイベントハンドラで宣言する．Rei47の`heap[]`や`n`，Rei65の`table[][]，hand[]`などは関数の外で宣言する．

該当プログラム

Rei34，Dr34_1，Dr34_2，Dr35_2，Dr37，Rei40，Dr40，
Rei43，Dr43，Rei44，Dr44_1，Dr44_2，Rei45，Dr45_1，Dr45_2，Rei46，
Dr46
Rei47，Dr47，Rei48，Dr48_1，Dr48_2
Rei65，Dr65，Rei67

- head，tail，rootなどの初期化を行う．

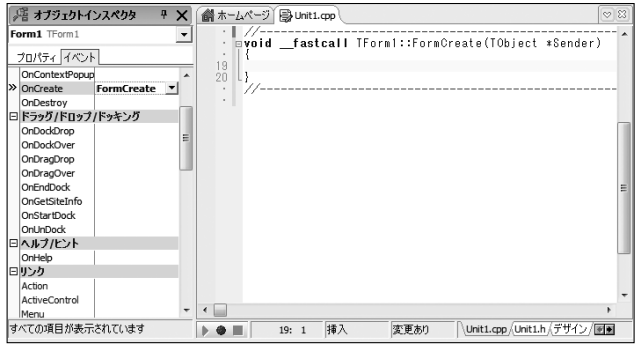
  該当プログラム

  Rei35，Dr35_1，Dr35_2，Rei36，Dr36，Rei37，Rei40，Dr40
  Rei44，Dr44_1，Dr44_2，Rei45，Dr45_1，Dr45_2，Rei46，Dr46
  Rei47，Dr47，Rei48，Dr48_1，Dr48_2

- 初期化処理をFormCreateイベントハンドラに置く．

  該当プログラム

  Dr34_2，Dr37，Rei41，Dr41



- ダミーノードを使用しているもので，登録終了後，再登録処理ができないもの．

  該当プログラム

  Dr36，De40

- ボタンを初めて押したときと2回目以後で処理を分けるために変数flagを導入．

  該当プログラム

  Dr34_1，Rei43，Dr43

- printf("¥n・・・・)のように書式制御文字列の先頭に¥nがあるものは単独のprintf("¥n")を置くか¥nを削除.

  該当プログラム

  Dr30_2，Dr32，Rei37，Dr37，Rei49

- 再度行うためにnを1に再初期化する.

  該当プログラム

  Rei48，Dr48_1，Dr48_2

入力ボックスとメッセージボックス

- ［OK］ボタンのメッセージボックス

```
MessageBox(NULL,"次の移動","",MB_OK);
```

メッセージにAnsiString型を使う場合はMessageBoxは使えないので以下

```
MessageDlg("次の移動",mtInformation,TMsgDlgButtons() << mbOK,0);
```

- ［はい］,［いいえ］ボタンのメッセージボックス

```
c=MessageDlg(AnsiString(q1->node),mtInformation,
TMsgDlgButtons() << mbYes << mbNo,0);
if (c==IDYES){      /* 子の接続 */
```

- 入力ボックス

```
ret=InputBox("","初期ノード？","");
ret.WideChar(root->node,30 );
```

retにAnsiString型の入力文字列が返されるのでWideCharでAnsiStringからwchar_t配列に変換

文字列型の問題

- Rei50において入力ボックス，メッセージボックスで扱う文字列の型がAnsiString型なので構造体メンバをchar配列からAnsiString型に変更した.

- Dr50_1，Dr50_2において動的メモリ割り当てを行う場合は固定長配列でなければならないのでAnsiStringの代わりにwchart_tを用いた．このためWideCharを使ってAnsiStringからwchart_t配列に変換する．AnsiStringコンストラクタでwchart_t配列からAnsiStringに変換を行う.

- ファイル出力

  ワイド文字のリード/ライトを行うのでfwscanfとfwprintfを使用する.

```
fwscanf(fp,L"%30ls%4d",p->node,&flag);
fwprintf(fp,L"%30ls%4d",p->node,Leaf);
```

  これに伴いロケールを地域ロケール（日本語）に設定しておく.

```
setlocale(LC_ALL,"");
```

  これに伴いwchar.h、locale.hをインクルードする.

Ⅴ TurboC++ GUI環境を使ったプログラムへの移植その2

printf関数を使わずGUI環境に合わせた出力方法を用いる．つまりⅣで示したxxxform.hは使用しない．ここではラベルまたはリストボックスへ出力を行う例を示す．

printfの書式制御はFormatメソッドまたはsprintf関数で行う．書式変換用にcbuf，1行出力用にoutstrという文字列型変数を用いる

コントロールはNameプロパティの名前（Label1やListBox1）でイベントハンドラ内で参照できる．ユーザ関数で参照するにはForm1->Label1やForm1->ListBox1とする．以下のようにして外部オブジェクトのLBを使って参照する方法もある．

```
TListBox* LB;
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    LB=ListBox1;
```

書式制御はsprintfでchar配列に出力しそれを，AnsiString()でAnsiStringに変換．

□スタティックテキスト（ラベル）への出力(サンプル：Dr1_2Label)

①ラベルのプロパティを以下のように設定する

- FontをMSゴシックのような固定フォントにする．
- ボーダースタイルは指定できない．

②Dr1_2Labelプログラムの主要部分の抜粋

```
char cbuf[100];          ←――― 書式変換用
AnsiString outstr;       ←――― 1行出力用

#include <stdio.h>
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    sprintf(cbuf,"%3ld   " ,combi(n,r));
    outstr+=AnsiString(cbuf);

    Label1->Caption=outstr;
}
```


□リストボックスへの出力

（サンプル：Dr14_2ListBox，Rei66ListBox，Dr66ListBox）

①リストボックスのプロパティの変更はない．

②Dr14_2ListBoxプログラムの主要部分の抜粋

```
AnsiString outstr;       ←――― 1行出力用
char cbuf[100];          ←――― 書式変換用
void printresult(short c[])        /* 結果の表示 */
{
    short i;
    for (i=0;i<L2;i++){
        sprintf(cbuf, "%04d",c[i]);
        outstr+=AnsiString(cbuf);
    }
    Form1->ListBox1->AddItem(outstr,NULL);
}
```

Ⅵ GCCの運用形態

MinGW（Minimalist GNU for Windows）Ver3.4.5をインストールすると以下のようなフォルダ構成となる．sourceフォルダはユーザが作成する．

コマンドパスに以下を追加しておく．

C:¥MinGW¥bin;C:¥MinGW¥libexec¥gcc¥mingw32¥3.4.5;

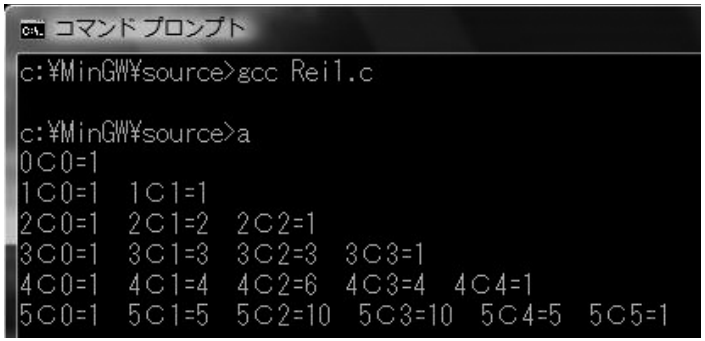
以下によりコンパイル・リンク，実行を行う．

>gcc Rei1.c

Rei1.cがコンパイル・リンクされ実行可能ファイルa.exeが生成される．

>gcc -o Rei1.exe Rei1.c

Rei1.cがコンパイル・リンクされ実行可能ファイルRei1.exeが生成される．



Ⅶ MinGW(Minimalist GNU for Windows)Ver3.4.5での注意点

対象プログラム： Dr17，8章（グラフィックス）以外のプログラム

- main関数の型

  main関数の型にvoidは認められないので以下の形式にする．

```
int main(void)
{

    return 0;
}
```

- 日本語

  シフトJISコードの下位バイトに0x5c（¥）を持つもの（表や能など）は，0x5cがエスケープ文字として使われてしまうので，対象となる漢字の後ろに¥を補う．コメント中の漢字は問題にならない．

```
printf("表¥が一杯です¥n");
```

  対象となる漢字として以下がある．

  ー ソ ｂl Ⅸ 噂 浬 欺 圭 構 蚕 十 申 曾 箪 貼 能 表 暴 予 禄 兎 喀 媾 彌 拿 杤 歔 濬 畚 秉 綵 臂 藹 觸 鱧 鐔 饅 鷭 偬 硲 纉 狄

対象プログラム： Dr25（表），Rei50（能）