はじめてのC++とROOT

内藤大輔

2007年8月19日

目 次

概 要

この説明書では、ROOT を使ったヒストグラムやグラフの作り方を説明します。ROOT とは C++でヒストグラムやグラフを簡単に描ける様にしたプログラムの集合です。ヒストグラムやグラフを作る作業は C++によるプログラミングになります。その為、最初の 1.1 章から 2.6 章までは C++の貴本的な機能を紹介します。3.1 章から 3.3 章で ROOT を用いたヒストグラムやグラフの作り方を説明します。途中、例題や練習問題が登場しますので、其の度にプログラムを書いて実行してみてください。

第1章　やってみよう

1.1　兎にも角にもやってみる

1.1.1　ROOTの起動

まず、パソコンの前に座り、端末を1つ開いて下さい。次にROOTを使えるようにします。次のコマンドを入力してください。但し、「>」はプロンプトですので入力する必要はありません。[Enter]はEnterキーを押すと言う意味です。

```
> source /usr/local/root/root_setup.sh [Enter]
```

これでROOTが使え様に環境設定が出来ました。次の様にしてROOTを実行してみましょう。

```
> root [Enter]
  ******************************************
  *                                        *
  *        W E L C O M E  to  R O O T      *
  *                                        *
  *   Version  5.14/00g       9 July 2007  *
  *                                        *
  *  You are welcome to visit our Web site *
  *          http://root.cern.ch           *
  *                                        *
  ******************************************

FreeType Engine v2.1.9 used to render TrueType fonts.
Compiled on 12 August 2007 for linux with thread support.

CINT/ROOT C/C++ Interpreter version 5.16.16, November 24, 2006
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0]
```

WELCOME to ROOT というメッセージが表示され、上記の様に root [0] というプロンプトが表示されれば成功です。.q と入力して Enter キーを押して下さい。ROOT が終了します。.q

1.1.2 Hello World!

では、C++のプログラムを書いて ROOT で実行してみましょう。プログラムを書くには emaca というソフトを使います。次の様にして hello.C というファイルを作りましょう。

```
> emacs hello.C &
```

次に示すサンプルプログラム hello.C を書いて保存して下さい。但し、左端の番号とコロンは説明の為の行番号です。プログラムを書くときは無視して下さい。

```
01: void hello()
02: {
03:   cout << "Hello World!" << endl;
04: }
```

２行目の {から４行目の } の間にプログラムを書きます。1 行目の void hello() の hello はファイル名と同じでなければなりません。この場合、ファイル名が hello.C なので void hello() になっています。また、例えば、ファイル名が test.C ならば 1 行目は void test() になります。

画面に文字を表示したいときは cout オブジェクトを使います。「＜＜」は cout オブジェクトに表示したいものを渡す記号です。３行目の”(ダブルクォート）から ” の間に表示したい文字を書きます。プログラムは基本的に上から下へと実行されていきます。この流れを制御するのが if 文と for 文です。これらは 2.4 章と 2.5 章で説明します。

３行目の cout から ;（セミコロン）までが 1 文です。文の最後には必ず ;（セミコロン）が付きます。プログラムの実行方法と実行結果は次の通りです。[okacdf2] /home/naitou/tmp ＞はプロンプトです。

```
[okacdf2] /home/naitou/tmp > root -l -q hello.C [Enter]
root [0]
Processing hello.C...
Hello World!
[okacdf2] /home/naitou/tmp >
```

1.1.3 練習

1. 自分の名前を画面に表示するプログラムを書け。その時、ファイル名は自分の名前にすること。例えば君が大輔という名前ならば、ファイル名は daisuke.C になる。

2. ＜＜ endl を取るとどうなるか？実行してみよ。

第2章　C++

2.1　変数と定数と演算

2.1.1　変数とその宣言

変数とは何らかの値を入れておける箱の事です。変数という箱には色々な種類があります。箱の中に入れるものによって箱の種類を使い分ける必要があります。この種類のことを変数の型と言います。

例えば、整数を入れる箱は整数型、実数を入れる箱は実数型の箱を用意しなければなりません。もちろん、文字列を入れる箱もあります。以下が変数の型の種類です。C++には、この他にも型の種類がありますが、今回の解析ではここにある型が使えれば十分です。

型の名前	入るもの
int	整数
double	実数
string	文字列
char	文字（1文字だけ）

次のプログラム、variables.C は、整数型 (int)、実数型 (double)、文字列型 (string) および文字型 (char) を宣言し使う例です。プログラムを書いて実行してみて下さい。

```
01: //
02: // Program name: variables.C
03: // 2007.8.17 Daisuke Naito
04: //
05: void variables()
06: {
07:   int    a;
08:   double b;
09:   string c;
10:   char   d;
11:
12:   a = 3;
```

```
13:   b = 3.14;
14:   c = "Hello World!";
15:   d = 'A';
16:
17:   cout << "int:    " << a << endl;
18:   cout << "double: " << b << endl;
19:   cout << "string: " << c << endl;
20:   cout << "char:   " << d << endl;
21: }
```

C++では、変数は使う前に必ず宣言します。プログラムの 7 行目から 10 行目が変数の宣言になっています。変数の名前はアルファベット、アンダーバーおよび数字を組合せて作ることができます。但し、名前の最初の 1 文字目はアルファベットかアンダーバーでなければなりません。それ以外の文字は特殊な意味を持ちますので使わ無いで下さい。

12 行目から 15 行目で宣言した変数に値を代入しています。17 行目から 20 行目で、cout オブジェクトを使って変数の値を画面に表示しています。

14 行目と 15 行目に注目して下さい。C++では、文字列と文字（1 文字だけ）を明確に区別します。ダブルクォートで囲むと文字列になり、シングルクォートで囲むと文字となります。ですから、次の様な代入はエラーとなります。

```
char d;
d = 'AB';
```

また、1 行目から 4 行目はコメントになっています。//から始まる行は無視されます。上の例の様にプログラムの始めに簡単な説明を付けておくと、あとから分りやすいです。

以下が variables.C の実行結果です。

```
[okacdf2] /home/naitou/tmp > root -l -q variables.C
root [0]
Processing variables.C...
int:    3
double: 3.14
string: Hello World!
char:   A
[okacdf2] /home/naitou/tmp >
```

2.1.2 変数の初期化

変数は初期化という形で値を代入することが出来ます。次のプログラム、variable2.C は初期化を用いて変数に値を代入しています。

```
01: //
02: // Program name: variables2.C
03: // 2007.8.17 Daisuke Naito
04: //
05: void variables2()
06: {
07:   int    a = 2;
08:   double b = 3.14;
09:
10:   cout << a << endl;
11:   cout << b << endl;
12: }
```

一般には、宣言した変数は上の例の様に初期化しておくのが安全だと言われています。みなさんも、宣言した変数は必ず初期化する様にしましょう。また、変数名は、その変数にどんな意味の値が入っているか、想像しやすい名前を付ける様にしましょう。

2.1.3 練習

1. variables.C を初期化による代入に変更せよ。ファイル名は variables3.C とせよ。

2. string 型の変数を 1 つ宣言し、それをまず、Hello World!で初期化し画面に表示せよ。その後、その変数に君の名前を代入して、画面に表示せよ。君がもし大輔という名前ならば次の様な実行結果が得られれば、この課題は完了だ。

```
[okacdf2] /home/naitou/tmp > root -l -q myname.C
root [0]
Processing myname.C...
Hello World!
Daisuke
[okacdf2] /home/naitou/tmp >
```

2.1.4 定数

定数はプログラムを読み易くするために使われます。次のプログラム (const_ex.C) はインチをセンチメートルに変換します。変換定数 c は変化することが無いので const 修飾子を付けて宣言しています。定数は初期化による値の代入しか出来ません。

```
01: //
02: // Program name: const_ex.C
03: // 2007.8.17 Daisuke Naito
04: //
05: void const_ex()
06: {
07:   const double c = 2.45; // in. to cm
08:
09:   double x = 2.0; // in.
10:
11:   cout << "Length: "  << x   <<  "in." << endl;
12:   cout << "Length: "  << c*x <<  "cm." << endl;
13: }
```

この例では const 修飾子の有用性を感じることは出来無いかもしれませんんが、今後多くの場面で登場する。徐々にその有用性を感じ取ることが出来ると思います。

2.1.5 演算

四則演算は簡単に行える。次のプログラム (ope1.C) を書いて実行してみて下さい。

```
01: //
02: // Program name: ope1.C
03: // 2007.8.17 Daisuke Naito
04: //
05: void ope1()
06: {
07:   int a = 2;
08:   int b = 3;
09:   int c = 0;
10:
```

```
11:   c = a + b;
12:   cout << "a + b = " << c << endl;
13:   c = a - b;
14:   cout << "a - b = " << c << endl;
15:   c = a * b;
16:   cout << "a * b = " << c << endl;
17:   c = a / b;
18:   cout << "a / b = " << c << endl;
19:   c = (a - b)*b;
20:   cout << "(a - b)*b = " << c << endl;
21:   c = a - b*b;
22:   cout << "a - b*b = " << c << endl;
23: }
```

演算の結合は算数と同じです。また、括弧を使えば結合を変更できます。この他にも C++には多くの演算子があります。必要に応じて紹介します。

2.1.6 文字列の結合

文字列は演算子「＋」を用いて次の様に行えます。次のプログラム (mojiretu.C) を書いて実行してみて下さい。

```
01: //
02: // Program name: mojiretu.C
03: // 2007.8.17 Daisuke Naito
04: //
05: void mojiretu()
06: {
07:   string a = "This is ";
08:   string b = "a pen.";
09:   string c = "a ball.";
10:   string d = "";
11:
12:   d = a + b;
13:   cout << d << endl;
14:   d = a + c;
15:   cout << d << endl;
16: }
```

2.1.7 練習

1. 円の面積を計算するプログラムを作れ。実数型の変数を 3 つ宣言し、それぞれ名前を r、pi、ans とせよ。r には半径を、pi には円周率を、ans には面積を代入せよ。また、pi は変化し無いので const 修飾子を使って宣言せよ。ファイル名は calc_circle_area.C とせよ。

2. 図 2.1 の様な中空の球体の側面（図の陰の部分）の体積を求めるプログラムを作れ。内側の半径を代入する変数の名前を r_inner、外側を r_outer とせよ。プログラム名は calc_ball_volume.C とせよ。

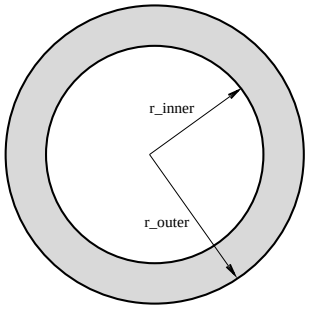

図 2.1: 中空の球体

2.2 数値を画面に表示する

ここでは数値を画面に表示する際に便利な技術を紹介します。
次の様にすれば、数値を 5 桁の幅で表示し、空いた部分は 0 で埋めます。

```
int a = 3;
cout.width(5);
cout.fill('0');
cout << a << endl;
```

次の例は小数を 10 の冪乗で表す方法です。

```
double b = 314.76;
cout.setf(ios_base::scientific, ios_base::floatfield);
cout << b << endl;
```

次の例は小数点 3 桁で四捨五入します。

```
double b = 314.76;
cout.setf(ios_base::scientific, ios_base::floatfield);
cout.precision(3);
cout << b << endl;
```

以上のコードを実行してみて下さい。

2.3 キーボードから数値や文字を入力する

2.3.1 cin オブジェクト

キーボードから文字や文字列、数値入力したいときは cin オブジェクトを使います。次の例 (myinput.C) はキーボードから文字列を入力し string 型の変数に代入し、画面に表示します。

```
01: //
02: // Program name: myinput.C
03: // 2007.8.17 Daisuke Naito
04: //
05: void myinput()
06: {
07:   string name = "";
08:
09:   cout << "Enter your name: ";
10:   cin >> name;
11:
12:   cout << "Your name is " << name << "." << endl;
13: }
```

2.3.2 練習

1. プログラム myinput.C を変更して数値を入力するようにせよ。そうするには変数を string 型から数値を代入できる型に変更せよ。変数名も name は止めて、数値らしいものにせよ。プログラム名は myinput_number.C にせよ。

2. 2 つの数値を入力し、その和を求め、結果を画面に表示するプログラムを書け。プログラム名は sum.C とせよ。

3. 2つの数値を入力し、その商を求めるプログラムを書け。プログラム名はshou.Cとせよ。このとき、数値を代入する変数の型はintが良いか、それともdoubleが良いか。

2.4 if文

2.4.1 プログラムの流れを制御するif文

今までの例からも分りますように、C++のプログラムは上から下へと流れて行きます。if文はある条件によってプログラムの流れを制御するために使われます。まずは具体的な例を見ていきます。次のプログラム (if_test1.C) を書いて実行してください。

```
01: //
02: // Program name: if_test1.C
03: // 2007.8.17 Daisuke Naito
04: //
05: void if_test1()
06: {
07:   int a = 3;
08:   int b = 2;
09:
10:   if (a>b) {
11:
12:     cout << "a>b is true." << endl;
13:   }
14:   else {
15:
16:     cout << "a>b is false." << endl;
17:   }
18: }
```

このプログラムは数値aとbの大小関係によって実行する文を分けています。もし、a＞bが真であるならば12行目が実行されプログラムは終了します。そうでなければ、すなわちa＞bが真以外ならば16行目が実行され、プログラムは終了します。

「＞」の様な比較を行う演算しを比較演算子と呼びます。以下に他の比較演算子を示します。C++にはこの他にも演算子があります。

式	意味
a>b	a の方が b より大きければ真。
a<b	a の方が b より小ければ真。
a>= b	a の方が b より大きいか、または等しければ真。
a<= b	a の方が b より小さいか、または等ければ真。
a==b	a と b が等ければ真。
a!=b	a と b が等くなければ真。

また、「かつ」や「また」の比較演算は次ぎ様に行います。次の例は「かつ (and)」を行います。

```
if ((a>b) && (c>b)) {

  文。。。
}
```

上の例は、数値 a、b、c、d を比較している。a>b でありかつ c>d である場合に真となり、文が実行されます。「&&」が論理積 (and) を意味します。

次の例は「または (or)」を行います。

```
if ((a>b) || (c>b)) {

  文。。。
}
```

上の例は a>b であるかまたは c>d である場合に真となり、文が実行されます。「||」は論理和 (or) を意味しています。

2.4.2 練習

プログラム、if_test1.C の a と b をそれぞれ 5、10 に変更して 16 行目の文が実行されることを確認せよ。

2.4.3 else if

次の様な関数を考える。

$$f(x) = \begin{cases} 0 & (x \leq 0) \\ x & (0 < x \leq 1) \\ 1 & (x > 1) \end{cases} \tag{2.1}$$

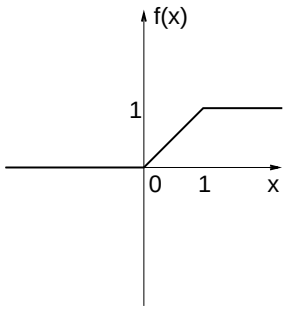

図 2.2: ある 1 次関数の振舞い。

次のプログラム（func.C）は上記の関数の振舞いを実現するために書かれたものです。

```
01: //
02: // Program name: func.C
03: // 2007.8.18 Daisuke Naito
04: //
05: void func()
06: {
07:   double x = 0.0;
08:
09:   cout << "Enter real number: ";
10:   cin >> x;
11:
12:   if (x<=0) {
13:
14:     cout << 0.0 << endl;
15:   }
16:   else if ((0.0<x) && (x<=1.0)) {
17:
18:     cout << x << endl;
19:   }
20:   else if (x>1.0) {
21:
22:     cout << 1 << endl;
23:   }
24:   else {
25:
```

```
26:     cout << "ERROR" << endl;
27:   }
28: }
```

x に値が入力されると、まず、12 行目の if 文で比較が行われます。この比較演算が真であれば 14 行目の文が実行され 28 行目に飛び、プログラムが終了します。もし、12 行目の比較演算が負であれば 14 行目の文は実行されず、16 行目の評価が始まります。ここで真となればそれ以降の else if 文は実行されません。最終的にどの条件も満さなければ 24 行目の else 文にたどりつき 26 行目の文が実行されます。

2.5 for 文と break

2.5.1 for 文

for 文は決められた回数だけ処理を実行したいときに用います。0 から 9 までの整数を画面に表示したいと考えてみましょう。次のプログラム (print_int.C) はそれを実行しています。

```
01: //
02: // Program name: print_int.C
03: // 2007.8.18 Daisuke Naito
04: //
05: void print_int()
06: {
07:   cout << 0 << endl;
08:   cout << 1 << endl;
09:   cout << 2 << endl;
10:   cout << 3 << endl;
11:   cout << 4 << endl;
12:   cout << 5 << endl;
13:   cout << 6 << endl;
14:   cout << 7 << endl;
15:   cout << 8 << endl;
16:   cout << 9 << endl;
17: }
```

これでも目的は達成していますが、for 文を使えばもっと簡単に書くことができます (print_int2.C)。

```
01: //
02: // Program name: print_int2.C
03: // 2007.8.18 Daisuke Naito
04: //
05: void print_int2()
06: {
07:   const int nmax = 10;
08:
09:   int i = 0;
10:   for (i=0; i<nmax; ++i) {
11:
12:     cout << i << endl;
13:   }
14: }
```

print_int2.C では、変数 i が 0 から始まります。10 行目の { から 13 行目の } までを実行する毎に変数 i に 1 加算されます。そうして、i<nmax を満す間、10 行目から 13 行目の間の処理を繰り返します。i の様な変数の事を制御変数と呼びます。

for 文の書式は次の様になっています。

```
for (変数の初期化; 条件; 変数の更新) {

  文
}
```

変数の初期化で制御変数の始まりの値をセットします。次に、条件を評価します。条件が真であれば変数の更新で制御変数の値を変化させます。print_int2.C では制御変数 i が 1 だけ加算されています。変数の初期化は最初の 1 回だけ行われます。

2.5.2 練習

1. 3 から 15 までの整数を画面に表示するプログラムを作れ。プログラム名は print_int3.C とせよ。また、for 文を使う際、条件に注意せよ。
2. 1 から 10 までの整数の和を求めるプログラムを作れ。
3. 1 から n までの整数の和を求めるプログラムを作れ。n は cin オブジェクトを用いて入力できる様にせよ。

4. 九九の表を画面に表示せよ。次の様に for 文を 2 重に使えば出来るだろう。

```
for (i=0; i<nmax; ++i) {

  for (j=0; j<nmax; ++j) {

    cout << i*j << endl;
  }
  cout << endl;
}
```

2.5.3 break

break は for 文を抜けたいときに使います。次の例は制御変数 i が 3 になったら for 文を抜けます。

```
for (i=0; i<100000; ++i) {

  if (i==3) {

    break;
  }
  cout << i << endl;
}
```

上記のプログラムを完成させて実行してみてください。break を体感してください。

2.6 ファイルからデータを取り出す

今回の o-Ps 寿命測定で得られるデータはテキストファイルで保存されます。テキストファイルとは emacs などで開いて人間が読むことのできるファイルです。ここでは、そのファイルからデータを読み込む方法を説明します。

ファイルからデータを読み込むには ifstream オブジェクトを使います。次の例 (file_test.C) は、ファイル (file_test.dat) に書かれた 1 列の数字を読み込む例です。

```
01: //
```

```
02: // Program name: file_test.C
03: // 2007.8.18 Daisuke Naito
04: //
05: void file_test()
06: {
07:   const int nmax = 10000;
08:
09:   ifstream data("file_test.dat");
10:   int      i = 0;
11:   int      x = 0;
12:
13:   for (i=0; i<nmax; ++i) {
14:
15:     if (data.eof()==true) {
16:
17:       break;
18:     }
19:     data >> x;
20:     cout << x << ' ';
21:   }
22:   cout << endl;
23:   data.close();
24: }
```

eof とは end of file の略で、ファイルの終端を意味します。このプログラムではファイルの終りまできたら for 文を抜けてファイルを閉じて、プログラムを終了します。

file_test.C に読み込ませるファイル file_test.dat は次のようなものを用意します。emacs で作ってください。

```
1
2
3
2
1
3
56
36
34
23
```

```
23
2
```

もし、データが 2 列あるようなものを読み込みたいなら、次のようにします。

```
data >> x >> y;
```

2 列のデータとは例えば次のようなものです。列と列の区切りは 1 つ以上の空白です。

```
1 34
2 3
23 54
12 4
```

2.6.1 練習

この節で紹介したプログラムを書いて実行せよ。2 列のデータを読み込むものや 5 列のデータを読み込むものを作ってみよ。これは、o-Ps のデータを解析するときに必要となる技術だ。

第3章　ROOT

3.1　ヒストグラム

ここからはROOTの機能を使っていきます。newなど新しい記号が登場しますが、難しく考えずやってみて下さい。

ヒストグラムを作るにはTH1Fオブジェクトを使います。次のプログラム(histo1.C)は、先程作ったfile_test.datから数値を詠み込み、それらをヒストグラムに詰めます。

```
01: //
02: // Program name: histo1.C
03: // 2007.8.18 Daisuke Naito
04: //
05: void histo1()
06: {
07:   const int nmax = 10000;
08:
09:   int    i = 0;
10:   double x = 0.0;
11:
12:   const int    nbin = 20;
13:   const double xmin = 0.0;
14:   const double xmax = 100.0;
15:   TH1F * h1 = new TH1F("h1", "Title", nbin, xmin, xmax);
16:
17:   ifstream data("file_test.dat");
18:   for (i=0; i<nmax; ++i) {
19:
20:     if (data.eof()==true) {
21:
22:       break;
23:     }
24:     data >> x;
25:     h1->Fill(x);
```

```
26:   }
27:   data.close();
28:   h1->Draw();
29: }
```

上の例の 15 行目に注目してください。xmin が x 軸の最小値、xmax が最大値です。nbin が xmin から xmax の間の分割数です。h1 はオブジェクト名です。また”Title”は、ヒストグラムの題名ですので好きな題名を付けてください。24 行目でファイルから読み込んだ数値を 25 行目でヒストグラムに詰め込みます。このプログラムはファイルの終端に到達するか、i が nmax になれば終了します。

実行結果は次の様になります。図 3.1 の様なヒストグラムが表示されれば成功です。但し、実行するときに -q を付け無いで下さい。

```
[okacdf2] /home/naitou/tmp > root -l histo1.C
root [0]
Processing histo1.C...
<TCanvas::MakeDefCanvas>: created default TCanvas with name c1
root [1]
```

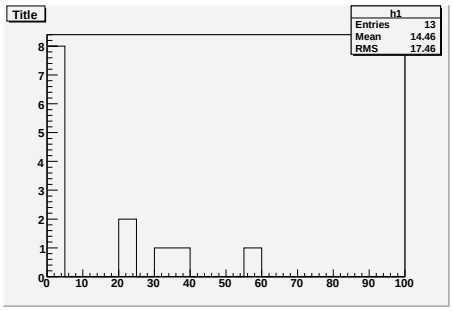

図 3.1: プログラム histo1.C の実行結果。

3.2 配列

配列はグラフを作るときに必要となります。
配列とは同じ型の変数が番号順に並んでいるものです (図 3.2)。
例えば、int 型の配列を 10 個宣言するには次の様に書きます。

```
int x[10];
```

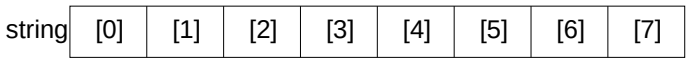

図 3.2: 配列とは変数を番号順に並べたもの。番号は 0 から始まる。

この場合、次の図の様に 10 個の int 型の配列が宣言されたことになります。

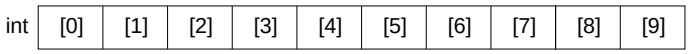

図 3.3: int 型の配列。

注意しなくてはならないのは、番号の付け方です。10 個の配列を宣言した場合、番号は 0 から順に 9 番まで割当られます。

また、配列の各要素を参照する場合は、[] を用いて次の様に書きます。

```
01:  x[1] = 0;
02:  cout << x[2] << endl;
```

1 行目は配列の 1 番要素に 0 を代入しています。2 行目は配列の 2 番要素を画面に表示しています。

次の例 (hairetsu.C) は 10 個の double 型配列を宣言し、それら全てに 0.0 を代入します。

```
01: //
02: // Program name: hairetu.C
03: // 2007.8.19 Daisuke Naito
04: //
05: void hairetu()
06: {
07:   const int nmax = 10;
08:
09:   int    i = 0;
10:   double x[nmax];
11:   for (i=0; i<nmax; ++i) {
12:
13:     x[i] = 0.0;
14:     cout << i << ": " << x[i] << endl;
15:   }
16: }
```

配列番号が 0 から 9 番であることに注意してください。

3.2.1 練習

20 個の int 型配列を宣言し、1 から 20 を代入せよ。それから、1 から 20 の和を求めよ。

3.3 グラフ

グラフを描くには TGraphErrors オブジェクトを用います。次の例 (test_graph.C) は test.dat ファイルからデータを読み込みグラフを作ります。

```
01: //
02: // Program name: test_graph.C
03: // 2007.8.19 Daisuke Naito
04: //
05: void test_graph()
06: {
07:   const int nmax = 10;
08:
09:   double x[nmax];
10:   double y[nmax];
11:   double xerr[nmax];
12:   double yerr[nmax];
13:   int    n = 0;
14:
15:   ifstream data("test_graph.dat");
16:   if(data.is_open()==false) {
17:
18:     cout << "Error: Can't open file: test_graph.dat." << endl;
19:   }
20:   else {
21:
22:     int i = 0;
23:     for (i=0; i<nmax; ++i) {
24:
25:       if (data.eof()==true) {
26:
27:     break;
```

```
28:       }
29:       data >> x[i] >> y[i] >> xerr[i] >> yerr[i];
30:       cout << i << " : " << x[i] << " " << y[i] << " "
31:        << xerr[i] << " " << yerr[i] << endl;
32:     }
33:     n = i;
34:     data.close();
35:
36:     TGraphErrors * gr = new TGraphErrors(n, x, y, xerr, yerr);
37:     gr->SetTitle("Test graph");
38:     gr->GetXaxis()->SetTitle("x-axis");
39:     gr->GetYaxis()->SetTitle("y-axis");
40:     gr->SetMarkerStyle(21);
41:     gr->Draw("AP");
42:   }
43: }
```

読み込むデータは次のようなファイル (test_graph.dat) になります。1 列目が x 軸の値、2 列目が y 軸の値、3 列目が x 軸のエラーの値、4 列目が y 軸のエラーの値になります。

```
1 20.2 0 2
2 30.5 0 2
3 40.1 0 2
4 50.9 0 2
5 60.2 0 2
6 70.5 0 2
7 80.7 0 2
8 90.2 0 2
9 100.4 0 2
10 110.7 0 2
```