「計算機入門及び演習」 C言語 — 待ち時間限定入力ルーティン

Dec. 6, 2000, 奥乃, *okuno@is.noda.sut.ac.jp, http://csai03.is.noda.sut.ac.jp/*

```
#include <stdio.h>
#include <math.h>
#include<fcntl.h>
#include<sys/time.h>
#include<sys/types.h>
#include<unistd.h>

int isready(int fd, int timeout)
{
  int rc;
  fd_set fds;
  struct timeval tv;

  FD_ZERO(&fds);
  FD_SET(fd,&fds);
  tv.tv_sec = timeout;
  tv.tv_usec = 0;

  rc = select(fd+1, &fds, NULL, NULL, &tv);
  if (rc < 0)
    return -1;

  return FD_ISSET(fd,&fds) ? 1 : 0;
}

double readarg(char *name, double defv)
{
  double value;
  int fd, x;

  fd = STDIN_FILENO;  /* stdin の fileno */

  printf("\n%sを入力して下さい: ", name);
  x = isready(fd, 5);
  if (x < 0) {
     exit(1); /* エラー */
  } else if (0 < x) {  /* 入力あり */
     scanf("%lf", &value);
  } else /* 時間切れ */ {
     printf("\007%.3f にしました!\007\n", defv);
     value=defv;
  }
  return value;
}
```

select - システム・コール ： 同期型入出力の多重化

【形式】

```
#include <sys/types.h>
#include <sys/time.h>

int select (width, readfds, writefds, exceptfds, timeou
int width;
fd_set *readfds, *writefds, *exceptfds;
struct timeval *timeout;
```

マクロ

```
FD_SET (fd, &fdset)
FD_CLR (fd, &fdset)
FD_ISSET (fd, &fdset)
FD_ZERO (&fdset)
int fd;
fd_set fdset;
```

【機能説明】

select() は, アドレスが readfds, writefds, exceptfds に引き渡される入出力記述子設定を調べ, それらの記述子のうち, 読み取り準備ができているもの, 書き込み準備ができているもの, または例外条件の待ち状態のものであるかどうかを検査します. width は, ファイル記述子を表す各ビット・マスクで, チェックすべきビット数です. 記述子セットの中で 0 から width -1 までの記述子が調べられます. 通常, width には, ファイル記述 子の最大数として, ulimit(3C) が返す値が入っています. select() が戻る際, 与えられた記述子セットは, 要求された処 理に対して準備のできている記述子から構成されているサブセットで置き換えられます. すべての設定の中で, 準備のできている記述子の総数が返されます.

timeout が NULL ポインタでない場合, timeout は, 選択の終了を待つための最大の時間間隔を指定します. timeout が NULL ポインタの場合, select は, 無期限にブロックします. ポーリングを実施するためには, timeout 引数は NULL 以外のポインタで, 値がゼロになっている timeval 構造体を指すものでなければな りません.

【timeval 構造体】 定義は /usr/include/sys/time.h

```
struct timeval
long tv_sec; /* seconds */
long tv_usec; /* and microseconds */
;
```

```
main()
{
  double a, b, c, d, x1, x2;

  a = readarg("x^2 の係数 ", 2.0);
  b = readarg("x の係数 ", 5.0);
  c = readarg("定数項 ", 3.0);

  printf("\n2 次方程式 \" %.3f x^2 + %.3f x + %.3f = 0 \" を解くと , \n 根は次のようになります \n", a, b, c);
  d = b * b - 4 * a * c;
  d = sqrt(d);
  x1 = (-b + d) / (2 * a);
  x2 = (-b - d) / (2 * a);
  printf("\tx1 = %f\n", x1);
  printf("\tx2 = %f\n", x2);
}
```

1 改良版の 2 次方程式解法プログラム

2 次方程式の根を求めるプログラムは上図のようになります.

1. これを重根や複素数の根が求められるように改良するとともに, 出力も工夫してみよう.
2. 同じ計算は 2 度しない等の高速化の工夫も取り入れてみよう.
   例えば, 上記のプログラムは平方根は 1 回しか計算していない！

2 計算機内での数の表現

2.1 計算機内での整数の表現

		4 ビットで 3	-3
1.			
2.			
3.			
4.			

2.2 IEEE 754 標準浮動小数点表現

32 ビット表現

1 ビット	8 ビット	23 ビット
符号 s	指数 e	仮数 m

は $(-1)^s \times (1+m) \times 2^{(e-127)}$ を表現.

$$\begin{cases} e = 255, m \neq 0: & \text{非数}(NaN) \\ e = 255, m = 0: & (-1)^s \times \infty \\ 0 < e < 255: & (-1)^s \times 2^{e-127} \times \underline{1.m} \\ e = 0, m \neq 0: & (-1)^s \times 2^{-126} \times \underline{0.m} \\ e = 0, m = 0: & (-1)^s \times 0 \end{cases}$$

64 ビット表現

1 ビット	11 ビット	52 ビット
符号	指数	仮数

は $(-1)^s \times (1+m) \times 2^{(e-1023)}$ を表現.

$$\begin{cases} e = 2047, m \neq 0: & \text{非数}(NaN) \\ e = 2047, m = 0: & (-1)^s \times \infty \\ 0 < e < 2047: & (-1)^s \times 2^{e-1023} \times \underline{1.m} \\ e = 0, m \neq 0: & (-1)^s \times 2^{-1022} \times \underline{0.m} \\ e = 0, m = 0: & (-1)^s \times 0 \end{cases}$$

2.3 IEEE 754 での例外処理

- 桁あふれ ($overflow$) : 演算結果が決められたビット数で表現できず.
- アンダーフロー ($underflow$) : 値の絶対値が小さすぎて表現できず.