1　ディスプレイにデータを出力する　30 頁

1.1　学習のポイント

printf を用いて整数、実数データを出力する方法を学びます。

1.2　整数データの出力

整数型変数 a の内容をディスプレイに出力するには次のようにします。(もし a の内容が 56 ならディスプレイに 56 が表示されます。)

```
printf("%d", a);
```

ところで、文字列と数値を混ぜて表示したいことはよくあります。たとえば、何かの合計を求めて表示するような時です。こうした場合は次のようにします。

```
printf("gokei=%d", a);
```

一般に、二重引用符 (") で囲まれた文字列を書式制御文字列と呼び、その後にカンマ (,) で区切って並べたものを引数 (ひきすう) と呼びます。書式制御文字列の文字は、そのままディスプレイに出力されますが、% に続く文字は、それに対応する引数の内容を表示する時の書式制御に使われ、それ自身はディスプレイに表示されません。もし上の例で a の値が 56 とすれば出力結果は次のようになります。

```
printf("gokei=%d", a);

gokei=56
```

1.3　例題 1 reidai1.c 32 頁

整数データ a と b の加算を行い、結果を表示しなさい。

```
/*
  整数データ a と b の加算を行い、結果を表示しなさい。
  reidai1.c
```

```
*/

#include <stdio.h>

int main()
{
  int a;
  int b;
  int x;

  a = 50;
  b = 20;
  x = a + b;

  printf("tasu = %d\n", x);

  return 0;
}
```

1.4 練習問題 1 rensyu1.c 32 頁

実数データ a と b の加減乗除を行い、結果を表示するプログラムを作りなさい。

```
/*
  実数データ a と b の加減乗除を行い、結果を表示するプログラムを作りなさい。
  rensyu1.c
*/

#include <stdio.h>

int main()
{
  float a;
  float b;

  a = 50.0;
  b = 20.0;

  printf("tasu   = %f\n", a + b);
```

```
  printf("hiku   = %f\n", a - b);
  printf("kakeru = %f\n", a * b);
  printf("waru   = %f\n", a / b);

  return 0;
}
```

2　機械学習と深層学習

2.1　蟻コロニー最適化法　群知能

蟻コロニー最適化法は、複数地点を巡る際の最短距離を求めるという趣旨の「巡回セールスマン問題」への適用において、良好な性能を示すことで知られています。蟻コロニー最適化法の原理は、蟻の群れが巣穴と餌場の間の最短距離を求める挙動に基づいています。巣穴と餌場の間に距離の異なる複数の経路があったとしましょう。蟻は大局的な判断はできませんから、どの経路が近道なのかを直接知ることはできません。そこで、最初は、蟻はランダムに経路を選択して移動します。

移動に際して、蟻はフェロモン (pheromon) と呼ばれる化学物質を分泌しながら移動います。ここで、フェロモンには蟻を惹きつける作用があるものとします。フェロモンによって、蟻は他の蟻が以前に通過した経路を選択しやすくなります。

フェロモンは経路上に蓄積しますが、揮発性の物質なので、時間とともに蒸発していまいます。距離の異なる経路がある場合では、往来が頻繁になりフェロモンが蓄積しやすくなります。

蟻の経路選択はフェロモンの濃度に影響を受けるので、当初ランダムだった経路選択は、だんだんとフェロモンの濃度の高い近道の経路の選択に偏ってきます。するとますます、近道の経路のフェロモン濃度があがります。このようにして、蟻の群れは距離の短い経路を選択するようになります。

機械学習と深層学習　小高知宏著　オーム社 78 頁

2.2　蟻コロニー最適化法 (aco) プログラム　aco により最適値を学習します aco.c

```
/****************************************/
/*                aco.c                 */
/*  蟻コロニー最適化法（aco）プログラム    */
```

```
/*    aco により最適値を学習します          */
/*使い方                                    */
/*aco                                       */
/*****************************************/


/*Visual Studio との互換性確保 */
#define _CRT_SECURE_NO_WARNINGS

/*ヘッダファイルのインクルード*/
#include <stdio.h>
#include <stdlib.h>

/*    記号定数の定義                        */
#define NOA 10 /*蟻の個体数*/
#define ILIMIT 50 /*繰り返しの回数*/
#define Q 3 /*フェロモン更新の定数*/
#define RHO 0.8 /*蒸発の定数*/
#define STEP 10 /*道のりのステップ数*/
#define EPSILON 0.15 /*行動選択のランダム性を決定*/
#define SEED 32768 /*乱数のシード*/


/*    関数のプロトタイプの宣言        */
void printp(double pheromone[2][STEP]) ;/*表示*/
void printmstep(int mstep[NOA][STEP]) ;
                                /*蟻の挙動の表示*/
void walk(int cost[2][STEP]
          ,double pheromone[2][STEP]
          ,int mstep[NOA][STEP] ) ;/*蟻を歩かせる*/
void update(int cost[2][STEP]
         ,double pheromone[2][STEP]
        ,int mstep[NOA][STEP] ) ;/*フェロモンの更新*/
double rand1() ;/*0〜1 の実数を返す乱数関数*/
int rand01() ;/*0 又は 1 を返す乱数関数*/


/**********************/
/*     main() 関数     */
/**********************/
```

```
int main()
{
 int cost[2][STEP]={/*各ステップのコスト（距離）*/
 {1,1,1,1,1,1,1,1,1,1},
 {5,5,5,5,5,5,5,5,5,5}};
 double pheromone[2][STEP]={0} ;/*各ステップのフェロモン量*/
 int mstep[NOA][STEP] ;/*蟻が歩いた過程*/
 int i;/*繰り返し回数の制御*/

 /*乱数の初期化*/
 srand(SEED) ;

 /*最適化の本体*/
 for(i=0;i<ILIMIT;++i){
  /*フェロモンの状態出力*/
  printf("%d:\n",i) ;
  printp(pheromone) ;
  /*蟻を歩かせる*/
  walk(cost,pheromone,mstep) ;
  /*フェロモンの更新*/
  update(cost,pheromone,mstep) ;
 }
 /*フェロモンの状態出力*/
 printf("%d:\n",i) ;
 printp(pheromone) ;

 return 0 ;
}

/*************************/
/*    update() 関数      */
/*    フェロモンの更新   */
/*************************/
void update(int cost[2][STEP]
        ,double pheromone[2][STEP]
        ,int mstep[NOA][STEP] )
{
 int m ;/*蟻の個体番号*/
 int lm ;/*蟻の歩いた距離*/
 int i,j ;
```

```
 double sum_lm=0 ;/*蟻の歩いた合計距離*/

 /*フェロモンの蒸発*/
 for(i=0;i<2;++i)
  for(j=0;j<STEP;++j)
   pheromone[i][j]*=RHO ;

 /*蟻による上塗り*/
 for(m=0;m<NOA;++m){
  /*個体 m の移動距離 lm の計算*/
  lm=0 ;
  for(i=0;i<STEP;++i)
   lm+=cost[mstep[m][i]][i] ;

  /*フェロモンの上塗り*/
  for(i=0;i<STEP;++i)
   pheromone[mstep[m][i]][i]+=Q*(1.0/lm) ;
  sum_lm+=lm ;
 }
 /*蟻の歩いた平均距離を出力*/
 printf("%lf\n",sum_lm/NOA) ;
}

/*************************/
/*    walk() 関数        */
/*    蟻を歩かせる       */
/*************************/
void walk(int cost[2][STEP]
,double pheromone[2][STEP],int mstep[NOA][STEP])
{
 int m ;/*蟻の個体番号*/
 int s ;/*蟻の現在ステップ位置*/

 for(m=0;m<NOA;++m){
  for(s=0;s<STEP;++s){
   /* ε -greedy 法による行動選択*/
   if((rand1()<EPSILON)
    ||(abs(pheromone[0][s]-pheromone[1][s])<1e-9))
   {/*ランダムに行動*/
    mstep[m][s]=rand01() ;
```

```
   }
   else{/*フェロモン濃度により選択*/
    if(pheromone[0][s]>pheromone[1][s])
      mstep[m][s]=0 ;
    else
      mstep[m][s]=1 ;
   }
 }
 }
 /*蟻の挙動の出力*/
 printmstep(mstep) ;
}

/*************************/
/*  printmstep() 関数      */
/*   蟻の挙動の表示        */
/*************************/
void printmstep(int mstep[NOA][STEP])
{
 int i,j ;

 printf("*mstep\n") ;
 for(i=0;i<NOA;++i){
  for(j=0;j<STEP;++j)
   printf("%d ",mstep[i][j]) ;
  printf("\n") ;
 }
}



/*************************/
/*    printp() 関数        */
/*   フェロモンの表示      */
/*************************/
void printp(double pheromone[2][STEP])
{
 int i,j ;

 for(i=0;i<2;++i){
  for(j=0;j<STEP;++j)
```

```
   printf("%4.2lf ",pheromone[i][j]) ;
  printf("\n") ;
 }
}

/*************************/
/*    rand1() 関数         */
/*0〜1 の実数を返す乱数関数*/
/*************************/
double rand1()
{
 /*乱数の計算*/
 return (double)rand()/RAND_MAX ;

}

/*****************************/
/*     rand01() 関数          */
/*     0 又は 1 を返す乱数関数 */
/*****************************/
int rand01()
{
 int rnd ;

 /*乱数の最大値を除く*/
 while((rnd=rand())==RAND_MAX) ;
 /*乱数の計算*/
 return (int)((double)rnd/RAND_MAX*2) ;

}
```

2.3 aco.exe の使い方

```
./aco
```

2.4 aco の結果

```
0:
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
*mstep
1 0 0 1 1 1 0 1 0 0
1 0 1 1 0 1 1 0 1 0
1 0 0 1 1 1 1 0 1 0
0 1 0 1 1 1 0 0 0 1
0 0 1 1 0 1 0 0 0 0
0 0 1 0 0 1 0 0 0 1
0 0 1 0 1 1 0 0 0 0
0 1 0 1 0 0 1 0 1 0
0 0 0 0 1 1 1 1 0 1
1 1 1 1 0 1 0 0 0 1
28.400000
1:
0.72 0.79 0.50 0.37 0.56 0.12 0.70 0.89 0.80 0.66
0.36 0.30 0.59 0.72 0.52 0.97 0.39 0.20 0.29 0.42
*mstep
0 0 1 0 1 0 1 1 0 0
1 0 1 0 0 1 0 0 1 0
0 0 0 0 0 1 1 1 1 0
0 0 0 0 1 0 1 1 1 0
1 0 0 1 0 1 0 1 1 0
0 1 1 0 0 0 1 0 1 1
1 0 1 1 1 1 0 0 1 1
0 0 0 1 1 1 0 0 1 1
1 1 0 0 0 0 1 1 1 1
0 1 1 1 1 1 1 1 0 0
30.400000
```