1　構造体へのポインタ　C 言語　164 頁

1.1　学習のポイント

構造体データを参照するためのポインタについて学びます。

構造体データを参照するためのポインタは、

```
struct eisei *p;
```

と宣言します。p が eisei 型構造体へのポインタ変数となります。このポインタ変数 p に先の構造体配列 a[] の先頭アドレスを次のように代入します。

```
p = a;
```

これにより、ポインタ変数 p を用いて次のように構造体配列 a[] を参照することができます。

つまり、

```
p->name  により a[0].name
p->kodo  により a[0].kodo
p->shuki により a[0].shuki
```

を参照します。
一般に、ポインタにより指し示されている構造体の各メンバの参照には、特別な演算子

```
->
```

を用いて、

ポインタ変数名->メンバー名

と書きます。
さて、ここで、ポインタ変数 p を、

```
p++;
```

としたとします。このとき

p->name **により** a[1].name
p->kodo **により** a[1].kodo
p->shuki **により** a[1].shuki

という次のデータを指すことになります。「C 言語」(河西朝雄著　ナツメ社)164 頁

1.2 **例題** 39 reidai39.c

練習問題 39　構造体配列 a[] のデータをポインタ変数 pa を用いて参照しなさい。

```
/*
  例題 39
  構造体配列 a[] のデータをポインタ変数 pa を用いて参照しなさい。
  reidai39.c
*/

#include <stdio.h>

int main()
{

  struct eisei{
    char *name;
    int  kodo;
    double shuki;
  };

  static struct eisei a[] = {{"Io", 5, 1.7691},
                             {"Europa", 6, 3.5512},
                             {"Ganymede", 5, 7.1545},
                             {"Callisto", 6, 16.6890},
                             {NULL, 0.0, 0}};

  struct eisei *pa;

  pa = a;
  while(pa->name != NULL){
    printf("%10s%3d%8.4f\n", pa->name, pa->kodo, pa->shuki);
    pa++;
  }
```

```
  return 0;
}
```

1.3 練習問題 39 rensyu39.c

例題 39 の構造体配列のデータを表示する関数 disp(pa) を作りなさい。関数 disp(pa) の pa には配列の先頭アドレスを渡すものとします。

```
/*
  練習 39
  例題 39 の構造体配列のデータを表示する関数 disp(pa) を作りなさい。
  関数 disp(pa) の pa には配列の先頭アドレスを渡すものとします。
  rensyu39.c
*/

#include <stdio.h>

struct eisei{
              char *name;
              int  kodo;
              double shuki;
};

void disp(struct eisei *pa)
{
  while(pa->name != NULL){
    printf("%10s%3d%8.4f\n", pa->name, pa->kodo, pa->shuki);
    pa++;
  }
}

int main()
{
  static struct eisei a[] = {{"Io", 5, 1.7691},
                             {"Europa", 6, 3.5512},
                             {"Ganymede", 5, 7.1545},
                             {"Callisto", 6, 16.6890},
```

```
                                    {NULL, 0.0, 0}};

    disp(a);

    return 0;
}
```

2　AI による大規模データ処理入門

2.1　遺伝アルゴリズムの実際

前回は、遺伝子プールの内容を表示するプログラム演習をしました。今回は、「AI による大規模データ処理入門」に沿って遺伝的アルゴリズムの実際をみていきましょう。

(前回の再掲)
簡単な例題を用いて、遺伝アルゴリズムの具体的な流れを説明します。今、次のような問題を考えます。

問題:数当てパズル

秘密にされた 8 桁の 2 進数がある。この 2 進数を推測し、秘密の 2 進数を求めよ。ただし、推測は何回でも行なえる。また推測結果に対して、何桁合っているかについてのヒントをその都度得ることができるものとする。

上記問題で、秘密の 2 進数は何でもかまいませんが、たとえばここでは、

秘密の 2 進数 (正解の値)　10101010

としておきましょう。

このパズルを解くには、推測値として適当な 8 桁の 2 進数を生成し、何桁合っているのかのヒントを頼りに、解をよりよくしていく必要があります。そこで、8 桁の 2 進数を遺伝子の表現として用いることにします。
AI による大規模データ処理入門　167 頁　小高知宏著

遺伝子の適応度は、遺伝子座を正解の値と比較した際に、正解と合致している部分の個数とします。たとえば「遺伝子プールの評価」のプログラムにより、0 番目の遺伝子は、4 箇所の遺伝子座が正解と合致しているため、適応度は 4 です。同じく 1 番目では、5 箇所が合致していますから適応度は 5 です。問題を解く過程では、どの遺伝子があっているかについての情報は与えられず、単に遺伝子の適応度のみが与えられるものとします。

遺伝的アルゴリズムでは、まず遺伝子の初期集団を生成します。最初は全くランダムに推測するしかありませんから、乱数を使って 0/1 の並びを作成します。前述のように遺伝子の集団を遺伝子プールと呼び、遺伝子プールに含まれる遺伝子の個数をプールサイズと呼びます。

上記において、カッコ内の値は各遺伝子の適応度です。初期集団の平均適応度は、3.75 となります。遺伝的アルゴリズムの目標は、遺伝子操作によって平均適応度を向上させることにあります。

手順に従い、遺伝子プールに対して遺伝子操作を施します。まず交叉により子遺伝子を作ります。そのためには、親となる遺伝子をルーレット選択など適当な手段により選び出します。ここでは、0 番目と 1 番目の親遺伝子が選ばれたものとします。
これらについて、交叉を実施します。交叉にはさまざまな方法がありますが、ここでは一点交叉を実施します。一点交叉は、遺伝子の適当な場所で遺伝子座の値を入れ替える交叉方法です。たとえば、例のペアについて、乱数によって交叉する点を選んだところ、右から 2 番目と 3 番目の遺伝子座の間で交叉を行うことになったとします。この場合、「交叉」プログラムのように子の遺伝子が生み出されます。

問題:遺伝子の初期集団 (遺伝子プール) の遺伝子の適応度を表示するプログラム (evalpool.c) を作成しましょう。

2.2 遺伝子プールの評価 evalpool.c

```
/************************************/
/*遺伝子プールの評価                */
/*evalpool.c                        */
/************************************/
#include <stdio.h>
int main()
{
 /*遺伝子プール*/
  int pool[4][8] = {{0,0,1,1,0,1,1,0},
                    {1,0,1,1,0,0,1,1},
                    {0,1,1,1,0,1,1,0},
```

```
                    {1,0,0,1,0,1,1,1}};

  int ans[8]  = {1,0,1,0,1,0,1,0};

  int    val;
  double sum;
  int    i;
  int    j;

  /*  プールの状態表示  */
  sum = 0;

  for(i=0;i<4;++i){
    val = 0;
    for(j=0;j<8;++j){
       if(pool[i][j] == ans[j]){
         val++;
       }

       printf("%d ",pool[i][j]);
    }
    printf(" (%d)\n", val);
    sum = sum + val;
  }

  printf("平均適応度 %f\n", (sum/4));

  return 0;
}
```

問題:一点交叉の例を表示するプログラム (cross.c) を作成しましょう。

2.3 遺伝子プールの評価 cross.c

```
/************************************/
/*交叉                              */
/*cross.c                           */
/************************************/
#include <stdio.h>
```

```
int main()
{
 /*遺伝子プール*/
  int parent[2][8] = {{0,0,1,1,0,1,1,0},
                      {1,0,1,1,0,0,1,1}};
  int ans[8]       = {1,0,1,0,1,0,1,0};

  int child[2][8];

  int pos;

  int sum;
  int val;
  int i;
  int j;


  /*  交叉  */
  for(pos = 0; pos < 8; pos++){
    if(pos < 6){
      child[0][pos] = parent[0][pos];
      child[1][pos] = parent[1][pos];
    }else{
      child[0][pos] = parent[1][pos];
      child[1][pos] = parent[0][pos];
    }
  }

  /* 親世代の遺伝子表示*/
  sum = 0;
  for(i=0;i<2;++i){
    printf("親 %d : ", i);
    val = 0;
    for(j=0;j<8;++j){
       if(parent[i][j] == ans[j]){
         val++;
       }

       printf("%d ",parent[i][j]);
    }
```

```
    printf(" (%d)\n", val);
    sum = sum + val;
  }

  printf("\n");

  /* 子世代の遺伝子表示*/
  sum = 0;
  for(i=0;i<2;++i){
    printf("子 %d : ", i);
    val = 0;
    for(j=0;j<8;++j){
       if(child[i][j] == ans[j]){
         val++;
       }

       printf("%d ",child[i][j]);
    }
    printf(" (%d)\n", val);
    sum = sum + val;
  }

  return 0;
}
```