

マージソートの実験

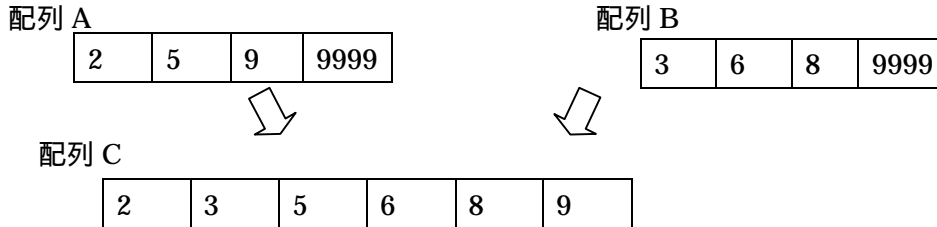
前のプリントと
差し替えてください。

マージソート処理実験用のクラスを作成して実験します。

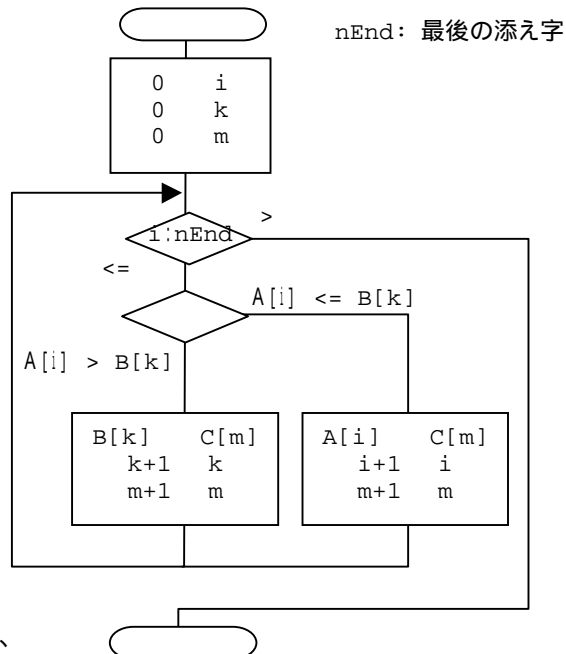
マージソートの基本的な考え方を示す。

既に整列された2つ部分を、マージ(併合)するイメージは次の通りである。

なお、9999 はデータでなく番兵で、これにより判定を簡単にしている。



上記配列 A と配列 B から配列 C を得る流れ図は、それぞれの配列をアクセスする添え字用の変数を i, k, m とすると、次のようになる。空欄を埋めよ



この考えでマージするわけですが、
 それにはソート済みの配列が、すでに出来上がっている場合の話です。
 そこで、配列を部分的に捉え、配列長が1つの場合のマージから始めます。
 これにより、配列長が2つの部分がたくさん出来上がります。
 出来上がった2個の部分配列をマージし、ソート済みで4個の部分配列を作ります。
 この出来上がった4の部分配列をマージして、8個の部分配列を作ります。
 ・・・・と行わせる再帰版のマージソートにするわけです。

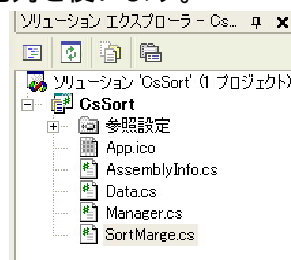
部分配列を再帰的に取り扱うので、流れ図のように3つの配列でなく、
 A と B を all で扱い、 C の配列の代わりに b の配列を使います。

作成ステップ プリント6ページまで出来上がっていて、

それに追加します。『プロジェクト』メニューから

『クラスの追加』で SortMarge クラスを追加します。

(ファイル名 SortMarge.cs)



```

01 using System;
02 namespace CsSort
03 {
04     //マージ ソート
05     public class SortMarge : Manager
06     {
07         Data []b; // 作業
08         public SortMarge(int n) : base(n)
09         {
10             b = new Data[n];
11         }
12         void mergeSort(Data []a, int nStart, int nEnd)
13         {
14             int nMid,i,k,m;
15             if(nStart < nEnd)
16             {
17                 nMid = (nStart + nEnd) /2;
18                 mergeSort(a,nStart, nMid);
19                 mergeSort(a,nMid+1 , nEnd);
20                 for(i = nMid+1; i > nStart; i--)
21                 {
22                     b[i-1] = a[i-1];
23                 }
24                 for(k = nMid; k < nEnd; k++)
25                 {
26                     b[nEnd + nMid - k] = a[k + 1];
27                 }
28                 // この時、 i==nStart で k==nEnd のはず!
29                 for(m = nStart; m <= nEnd; m++)
30                 {
31                     if(i <= nMid && b[i].points <= b[k].points)
32                     {
33                         a[m] = b[i++];
34                     }
35                     else
36                     {
37                         a[m] = b[k--];
38                     }
39                 }
40             }
41         }
42         public void Sort()
43         {
44             mergeSort(all,0,this.all.Length -1);
45         }
46         public static void Main()
47         {
48             SortMarge manage = new SortMarge(50000);
49             manage.Load("random.txt");//読み取り
50             int startTime = System.Environment.TickCount;
51             Console.WriteLine(
52                 "システム起動後からの経過時間：{0}ミリ秒", startTime);
53             manage.Sort();
54             int endTime = System.Environment.TickCount;
55             Console.WriteLine("システム起動後からの経過時間：{0}ミリ秒", endTime);
56             Console.WriteLine("整列作業の時間：{0}ミリ秒", endTime-startTime);
57
58             manage.Save("sort.txt");//保存
59
60             //manage.debugPrint();//デバック表示
61             Console.ReadLine();
62         }
63     }
64 }

```

マージ
してる
範囲

bの配列に
nStart ~ nMid が
昇順に並び、
nMid+1 ~ , nEnd
が降順になるように
コピーしている。
これで、番兵の代わり
になる壁を作っている。

安定な整列がで
きるように変更し
ました。

テキスト 67 の流れ図の通りに作成した Shell ソート

```
using System;

namespace CsSort
{
    //シェル法 の整列
    public class SortShell : Manager
    {
        public SortShell(int n) : base(n)
        {
        }
        public void Sort()
        {
            int k = this.all.Length;
            while(k > 1)
            {
                k = k/2;//kの間隔で、挿入または交換法を、k回使う。
                for(int m = 0; m < k; m++)
                {
                    for(int h = m + k; h < this.all.Length; h+=k)
                    { //挿入法の外側ループ
                        int j;
                        for(j = h-k; j >= 0; j-=k)
                        {
                            if(all[j].points > all[j+k].points)
                            {
                                Data w = all[j];
                                all[j] = all[j+k];
                                all[j+k] = w;
                            }
                            else
                            {
                                j = -1; // break でもよい
                            }
                        }
                    }
                }
            }
        }

        public static void Main()
        {
            SortShell manage = new SortShell(50000);
            manage.Load("random.txt");//読み取り
            int startTime = System.Environment.TickCount;
            Console.WriteLine(
                "システム起動後からの経過時間：{0}ミリ秒", startTime);
            manage.Sort();
            int endTime = System.Environment.TickCount;
            Console.WriteLine("システム起動後からの経過時間：{0}ミリ秒", endTime);
            Console.WriteLine("整列作業の時間：{0}ミリ秒", endTime-startTime);

            manage.Save("sort.txt");//保存
            //manage.debugPrint();//デバック表示
        }
    }
}
```

テキスト 71、72 の流れ図を参考にしたクイックソート

```

using System;
namespace CsSort
{
    //クイック ソート
    public class SortQuick : Manager
    {
        public SortQuick(int n) : base(n)
        {
        }
        // 分割
        int quickDiv(int nStart, int nEnd)
        {
            int i = nStart;
            int j = nEnd;
            int P = all[(i + j)/2].points;
            for(;;)
            {
                while(all[i].points < P) i++;
                while(all[j].points > P) j--;
                if(i>=j) return j;
                Data w = all[j];//交換
                all[j] = all[i];
                all[i] = w;
                i++;
                j--;
            }
        }
        void quickSort( int nStart, int nEnd)
        {
            if(nStart >= nEnd) return;
            int idx = quickDiv(nStart,nEnd);
            quickSort( nStart, idx);
            quickSort( idx+1, nEnd);
        }
        public void Sort()
        {
            quickSort(0,this.all.Length -1);
        }
        public static void Main()
        {
            SortQuick quick = new SortQuick(50000);
            quick.Load("random.txt");//読み取り
            int startTime = System.Environment.TickCount;
            Console.WriteLine(
                "システム起動後からの経過時間：{0}ミリ秒", startTime);
            quick.Sort();
            int endTime = System.Environment.TickCount;
            Console.WriteLine(
                "システム起動後からの経過時間：{0}ミリ秒", endTime);
            Console.WriteLine("整列作業の時間：{0}ミリ秒", endTime-startTime);
            quick.Save("sort.txt"); //保存
            //quick.debugPrint();//デバック表示
            Console.ReadLine();
        }
    }
}

```