FUNCTION COMPRESS－圧縮実験
Last Update 06/2/8

圧縮実験

VBのプロジェクトを更新しました。-06/2/8

Function Compress の圧縮シミュレーションをしましょう。前回の「Function Compressとは？」で、アルゴリズムの紹介は済ませてありますので、まだご覧でない方は、読んでみることをお勧めします。

今回は、「VC++6.0」で、コアのDLLをつくり、GUI部分を「VB6.0」でつくり、DLLの読み込みの形でVB側で圧縮結果を出力します。

VC++の方からまぜ説明します。

まず、しなくてはならないこと・・・それは、圧縮を開始する関数の用意です。「Compress」関数とでもしましょうか。これを定義します。シュミレーション用のファイルサイズの、前後を格納する構造体を、まずは宣言します。

```
struct ReturnFileInfo
{

   int BeforeSize;
   int AfterSize;

}; typedef ReturnFileInfo RFI;
```

次に、先ほどでてきた圧縮を制御するメインループの「Compress」関数です。内部でファイルの圧縮が終わるまで、for()でループさせてあります。

尚、VB側のDLL呼び出し規約は「_stdcall」です。以下のように定義をしましょう。

１:まずは「windows.h」のインクルード。
２:簡単にするため、"defファイル"の用意。
３:「FAR WINAPI」を用いて関数を定義。

こうすると、VB側で関数名でDLL関数を呼べるようになります。

さて、次に圧縮アルゴリズムとその関数。この規格は、圧縮のアルゴリズムとデータの格納が独特です。最高のメリットは、バージョンやアルゴリズムが変わっても、解凍アルゴリズムはまったく一緒です。極端な話、規格に合わせれば100年先でも解凍ができるわけです。その味噌は関数の式。式でデータを格納するのです。なので、まずは関数の式を定義します。今回の圧縮アルゴリズムは、一番簡単な"ランレングス"を用いることにしましょう。

an=a0+sn
sn=b

という階差数列の、階差が重要です。これを関数の式に置き換えると、

F(x)=ax

となります。xは、増分が一定です。ヘッダの定義により変更できます。今回は、シミュレーションということも考えてaと増分を可変にすることにしましょう。逆に言えば、線形性がある式なので、増分を1・乗数をaと考えることもできます。命令がincですむので、若干早いかも。根拠は無いですけどねぇ・・・。

プログラミングのほうに話を移します。

Compress関数の引数は、

１:圧縮するファイル名
２:圧縮データ数
３:増分(x ＊ t)
４:F(x)初期値
５」圧縮レベル

の順番です。返り値は、「RFI 構造体」です。

```
RFI FAR WINAPI Compress(char* filename,unsigned char MustTo,
                        unsigned char  IncrementSize,unsigned char SHOKI,
                        unsigned char CompressRevel)
{

    long OKSIZE=0;
    long NOSIZE=0;

    RFI RET;

    int hundle=open(filename,O_RDONLY); int FileSize=filelength(hundle);

    RET.BeforeSize=FileSize;

    unsigned char* Data=new unsigned char[FileSize];

    read(hundle,Data,FileSize);

    unsigned char FinData=Data[0];

    register TMPNO=0;

    register FinAdd=0;

    register i;

    for(i=0;i<=FileSize-MustTo;i++)
    {

        if (!   TestFunc(Data,FinData,i,i+MustTo,IncrementSize,7-TMPNO,SHOKI,CompressRevel)  )
        {

            TMPNO++;

            FinData=Data[i];

            if (TMPNO==7)
            {

                NOSIZE+=8;
                TMPNO=0;
```

```
            }
        }
        else
        {

            i+=MustTo;
            FinData=Data[i-2];

            NOSIZE+=TMPNO;
            OKSIZE++;
            TMPNO=0;
         }
    }

    RET.AfterSize=OKSIZE+NOSIZE;

    return RET;

}
```

次に、「Test Func」関数です。この関数は、圧縮レベルの設定に基づき、後述する「Function」関数を呼び、圧縮結果を

「True or False」

つまり、booleanで返します。これは言い換えれば圧縮成功かどうかを示します。この情報に基づき、Compressが圧縮データの配置を行います。ですが、今回は配置は抜きにして、圧縮結果のみを加算して、AfterFileSizeとして出力するようにしました。

引数は、圧縮するデータのunsigned charポインタ・開始の値・開始アドレス・終了アドレス・増加量・直前の圧縮不可数・初期値の0化・圧縮レベル の順です。

```
bool TestFunc(unsigned char* Data,unsigned char StartValue,
                long StartAdd,long EndAdd,unsigned char IncrementSize,
                unsigned char OKBit,unsigned char SHOKIYESNO,
                unsigned char CompressRevel)
{

    long OKSIZE=0;

    register i,u;

    unsigned char FuncInfo=0;

    long MustTo=EndAdd-StartAdd;

    for(i=0;i<=(OKBit-1);i++)
    {

        FuncInfo=(FuncInfo<<1)+1;

    }
            for(i=0;i<=FuncInfo;i+=CompressRevel)
    {
         if (OKSIZE==MustTo)return true;

        OKSIZE=0;

        u=0;

        while(u<=(MustTo-1))
        {
```

```
            u++;

            if (Data[StartAdd+OKSIZE]==Function(IncrementSize*
                                                (u-SHOKIYESNO),
                                                StartValue,
                                                OKBit,
                                                (i & FuncInfo)
                                                )
              )
            {
                  OKSIZE++;
            }
            else
            {
                  u=MustTo;
            }
        }
    }

    return false;

}
```

最後に、「Function」です。この関数は最重要関数です。関数の「項の定義」に基づき圧縮の可否を出力しますが、結果を「TestFunc」が判断し、項の全パターンでのシミュレーションをし、最良の結果を出力することができます。圧縮に成功するか否かは問わず、関数ブロックの情報に基づき、F(x)の値を出力します。

引数は　直前の値・開始値・項の利用状況・関数情報　です。

```
unsigned char Function(int LetValue,unsigned char StartValue,int OKBIT,unsigned char FuncInfo)
{

     register i,u;

    unsigned char VekiValue;

    for (i= 0;i<=OKBIT-1;i++)
    {

        if ( ( (FuncInfo<<(OKBIT-i)) >>7 ) == 1 )
        {
                VekiValue=LetValue;
                u=1;

                while (u<=i)
                {
                      u++;
                      VekiValue*=VekiValue;
                 }

                 StartValue+=VekiValue;

        }
     }

      return StartValue;

}
```

この3つをDLLに実装しています。このDLLを呼び出し、圧縮結果を出力するのがVBのプロジ

エクトです。

まず、外観をご覧ください。

シミュレーション図

上のソフトは、VBのプロジェクトです。左下はブロック情報設定・右側は、左側のブロックの設定の値を全パターン行った結果です。右のほうにある縦線が圧縮率100%線です。この規格の圧縮率は最高で114パーセントです。つまり膨張します。左に行くほど、圧縮性能がいいわけですよ。

ちなみに、上の図の場合、圧縮したファイルがこのソフト自体。圧縮率は50%でした。すっごくべらぼうに時間がかかります。はっきり言って、メガファイルの圧縮は徹夜作業です。今後は改善して行きたいですけど・・・。

ダウンロード

ReadMeはありません。著作権は放棄しませんが、オープンソースですので、配布はかまいません。ですが、このアルゴリズムにのっとったアルゴリズムの公開時、または事後、このアルゴリズム自体の著作権まで主張なさるのはおやめください。

VC++6.0プロジェクト

VB6.0プロジェクト