new-delete

new は、動的に新たなメモリ領域を確保する演算子です。
C 言語で言う malloc 関数に近いものと考えてください。
「動的に」とは、通常のようにソースに配列の要素数などを書き込んだりする（静的）のではなくプログラムが作動してからメモリ領域を確保してくる方法です。
これにより、必要なときに必要なメモリを必要なだけ取ることが出来るようになるのでより柔軟なプログラミングを行えます。

ポインタ変数 = new データ型;

delete は new で取得したメモリ領域の破棄を行います。
new で取ってきたメモリ領域は、必ず delete で破棄しなければいけません。でないとメモリ領域はパソコンの中に残り続け、それを繰り返すと最終的にはメモリをすべて使い切ってパソコン自体動かなくなるということもありえます。こちらは通常 Form のデストラクタに書き込みますが、関数内での new など、メモリ領域に必要が無くなれば消すといいでしょう。

delete ポインタ変数;

new []-delete []

delete[]とは配列の領域を破棄するものです。ただし 1 次元分しか消せないので 2 次元以上の場合、for 文などで行と列ごとに消す必要があります。

delete [] ポインタ変数;

delete[]が配列を消すなら、new[]は配列の領域を取得する、と考えた人もいるでしょうがそういうことにはなっていません。
配列のメモリ領域を取得するには、

ポインタ変数 = new データ型[要素数];

と書く必要があります。
要素数は変数名でもいいですが、何も書かないとエラーになるので注意してください。これも 1 次元分のみの領域を取得します。多次元配列の場合は for 分などの繰り返しを利用してください。

デストラクタ

デストラクタとはプログラムの終了時に呼び出されるもので、通常のプログラムでは必要ありません。しかし、プログラムの終了時に何か処理を追加したい場合などはこれを使います。

使い方

まず、ヘッダーの public の宣言の辺りに、関数の宣言のように次の文を書き込みます。

```
__fastcall ~TForm1();
```

次に cpp ファイルに移り、これまた関数のように書きます。

```
__fastcall TForm1::~TForm1()
{

}
```

後はこの中にプログラム終了時の処理を書き込むだけです。

※注意　デストラクタのヘッダーへの宣言ですが「__fastcall」は必ずいるので抜かさないようにしてください。また、cpp に書くときに「TForm1::」を書くのを忘れないようにして下さい。

new-delete の使用例

TstringList を使って説明します。これは複数の文字列をまとめて扱うためのものです。

まずポインタ変数を宣言しましょう。

```
TstringList *str;
```

これを new 演算子を使ってメモリ領域を当てはめます。

```
str = new TstringList;
```

これで、TstringList 型の str という変数の使用が可能になりました。

さて、str の使用が終わった、あるいはプログラムを終了する場合は取得したメモリ領域を開放する必要があるので

```
delete str;
```

とします。

プログラム終了時に開放するならデストラクタに書いておきましょう。