Charsets

So, what is charset I hear you ask. Charset consists of three files, Charset.BI, Charset.BAS and Testsets.BAS. The first 2 files are the interface and the source code for a small library that deals with an abstract data type while the final file is a very rough and ready test program for the library. The abstract data type that is dealt with is called a CharSet and is a simulation of mathematical sets in the domain of the 256 different characters of the PC's (western) character set (cor! What a mouthful! :-)). In other words, you now no longer need Pascal if you want to use sets of characters. In fact, with my implementation, you have 18 different things you can do with sets as opposed to the ten you can with Pascal.

If you know about set theory please skip to the next section because what I am about to say is for those who have no idea what I'm talking about up to this point. Beware what follows is a gross over-simplification and is the absolute minimum needed in order to make full use of the charset library.

What is a set? A set is a collection of things, with no duplicates. What sort of items. Well in our case characters (those things you get when you use CHR$() :-)). So, what is the minimum number of items needed for a collection of items to be a set? Absolutely zero (this is termed the empty set). The items that are part of a set are termed the members of the set and the number of members a set has is termed the cardinality of the set. Another term to do with sets is subset, which is itself a set. Suppose you have two sets. One set will be a subset of the other if all of its members are included in the other. By this definition the empty set is a subset of all other sets and a set is a subset of itself. There is another type of subset, called a strict (or proper) subset. This type of subset is identical to an ordinary subset except that a set cannot be a strict subset of itself. Five set operations that you need to know about are Union, difference, intersection and symmetric set difference and complement. The first 4 of these operations take 2 arguments and return a third.

Set union - The action of this operation is to combine the contents of the 2 arguments and place the result in the third. The order of the 2 arguments is not important.

Set difference - This operation removes those items contained in the second argument that also occur in the first argument, from the first argument and places the result in the third. The order of the first 2 arguments is paramount.

Set intersection - All the items that occur in both the first argument and the second argument are placed in the third. Order of the first 2 arguments is unimportant.

Symmetric set difference - All the items that occur in either the first argument or the second argument but do not occur in both arguments simultaneously are placed in the third. Order of the first 2 arguments is not a problem for this operation.

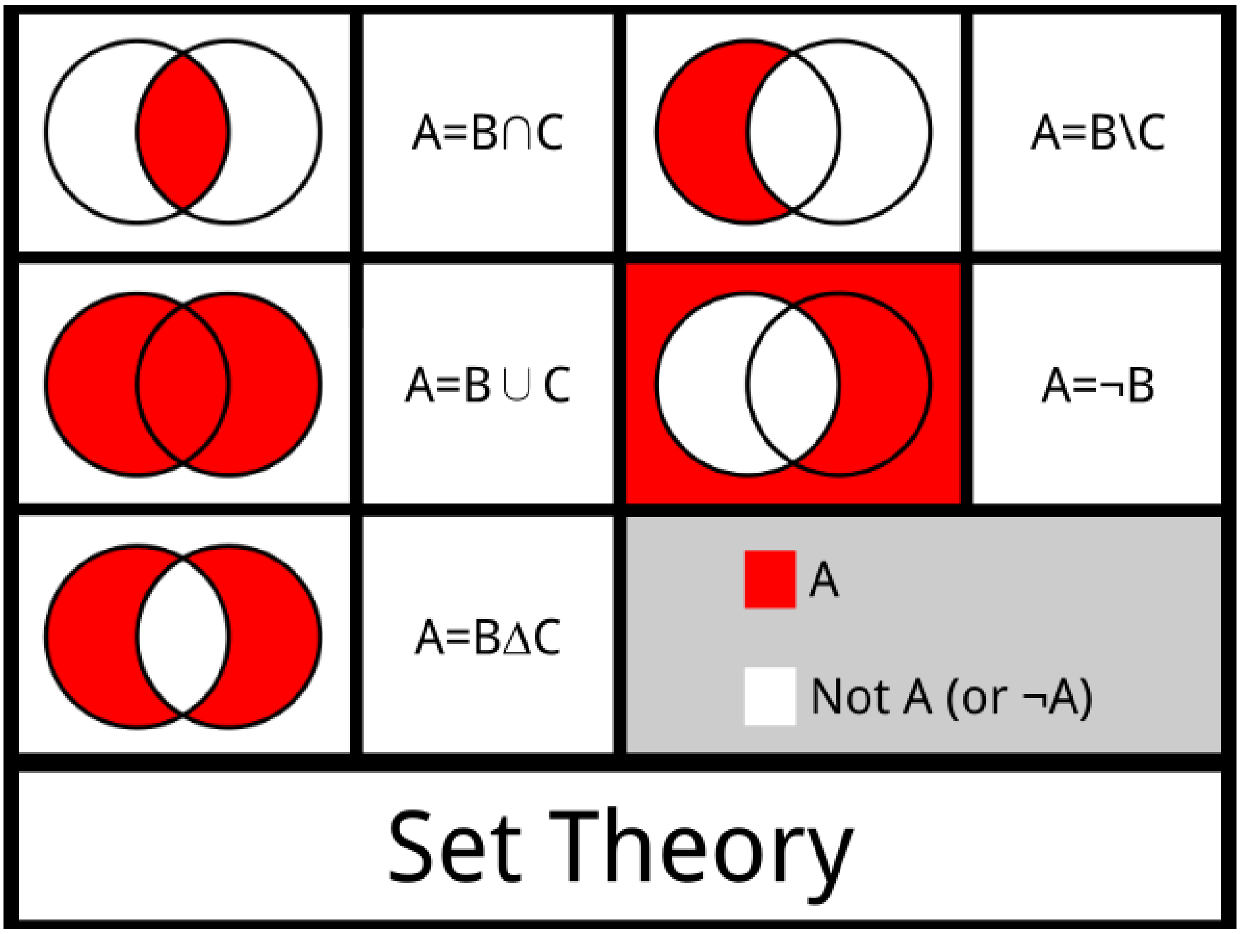
Set complement - This operation takes a single argument and returns a second. In order to explain what this operation does I will have to introduce another term. That term is "the universe of a set". The universe of a set is all the items that could possibly be held in a set. In the case of the library that I have placed in this echo that is all the characters in the PC's (western) character set i.e. CHR$(0) to CHR$(255) inclusive. What set complement does is to place into the second all the items in the universe of the set *except* those that are already in the first argument.

Having dealt with sets in general, I will now deal with the library I have written. As I have already said a number of times, the sets that are manipulated in this library deal exclusively with characters. This is not as great a limitation as might at first be imagined as (at least in Pascal) sets of characters are by far the most heavily used type of set. So, what can you do with these sets of characters? Below is a list of all the routines that are in the library. Those items marked with a P are also supported by Pascal, while those that are marked with an T are part of formal set theory. Those which remain are there to increase the usefulness of the library as a whole. For further information see the comments in CharSet.BI.

Description	Routine	Pascal/Theory
Initialise a set	`SUB InitialiseSet`	P
Add items to a set	`SUB IncludeInSet`	P
Remove items from a set	`SUB ExcludeFromSet`	P
Assign one set to another	`SUB CopySet`	P
Make a set completely empty	`SUB MakeSetEmpty`	
Copy the contents of a set to a string	`SUB GetSetContents`	
How many items are in a set	`FUNCTION Cardinality%()`	T
Is an item a member of set	`FUNCTION IsMember()`	PT
Is a set empty	`FUNCTION SetIsEmpty()`	
Are two sets identical	`FUNCTION SetEquality()`	PT
Are two sets not identical	`FUNCTION SetInequality()`	PT
Is one set a subset of another	`FUNCTION IsSubsetOf()`	PT
Is one set a strict subset of another	`FUNCTION IsStrictSubsetOf()`	T
Set Complement	`SUB SetComplement`	T
Set Union	`SUB SetUnion`	PT
Set Difference	`SUB SetDifference`	PT
Set Intersection	`SUB SetIntersection`	PT
Symmetric Set Difference	`SUB SymmetricSetDifference`	T

Set Theory

Because most people find that images can explain better than words on their own. Please regard the illustration a complement to the explanation in the first part of this readme. Here is a standard illustration of the operations with sets and their results.



In the above illustration sets are represented by circles. So, in each box we have 2 sets. The boxes themselves represent the universe in which the sets operate. In each case above, each operation has an illustration of an operation and its standard notation in the box on its immediate right. The meanings of the operators are as follows -

$\cap$ = Set Intersection
$\cup$ = Set Union
$\triangle$ = Symmetric Set Difference
$\setminus$ = Set Difference
$\neg$ = Set Complement

The areas in red show Set A will contain after each operation.