Structures Bigger than 32K

The following example illustrates a C programming technique that is not necessarily Macintosh-specific.

```
/*
 * Structures bigger than 32K
 * This is a very simple program to demonstrate using dynamically allocated structs
 * of larger than 32K in size.
 * This example will allocate a struct larger than 32k, then fill the array field of the
 * struct with some squares of numbers and  the char fields of the struct with some
 * chars, and then print out the contents of the struct.
*/

#include <stdio.h>
#include <stdlib.h>

#define myArraySize 4000

typedef struct {
	long *myarray;
	char aChar;
	char anotherChar;
					// other fields of the struct here;
	} myLargeStructType;

myLargeStructType  *myLargeStruct;

main()
{
	long	i;

	/*  first allocate struct */
	/*  and of course check for allocation getting done w/o error */
	if ((myLargeStruct = malloc(sizeof(myLargeStructType))) == NULL)
	{
		printf ("Unable to allocate memory for struct\n");
		exit(1);
	}

	/*  now allocate array field of struct */
	if ((myLargeStruct->myarray = (long *) calloc (myArraySize, sizeof (long)))
		== NULL)
	{
		printf ("Unable to allocate memory for array\n");
		exit (1);
	}

	/* now to see how to use the struct, fill array field with all squares of i */
	for (i = 0; i < myArraySize; i++)
	{
		myLargeStruct->myarray[i] = i*i;
	}

	/* now fill char fields of struct */
	myLargeStruct->aChar = 'a';
	myLargeStruct->anotherChar = 'b';


	/* now print out first 10 squares */
	for (i = 0; i < 10; i++)
		printf ("%ld %ld\n", i, myLargeStruct->myarray[i]);

	/* now print out char fields */
	printf ("%c  %c\n", myLargeStruct->aChar, myLargeStruct->anotherChar);

}
```