Two-Dimensional arrays

Implemented as an array of array,

```
#define N_STUDENTS 60
#define N_MODULES  5


int results[ N_STUDENTS ] [ N_MODULES ];


results[ s ][ m ] = 65;
```

Each element of a 2-D array is a 1-D array

Assuming printarr & average are defined as below, this section of code is legal:

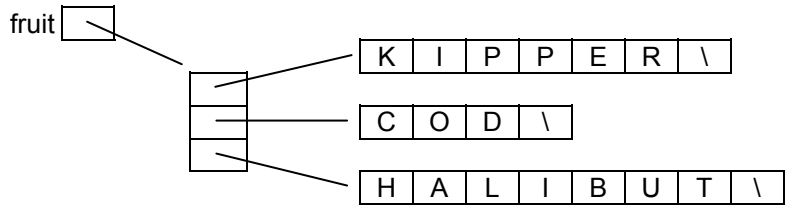
```
    for( s=0; s<N_STUDENTS; s++ )
    {   printarr( results[s], YMAX );
        printf( " ave = %i", average( results[s], YMAX ));
    }
```

```
void printarr( int *A, int size )
{    int i;
     printf( "\n" );
     for( i=0; i<size; i++ )
          printf( "%i, ", A[i] );
}

int average( int *marks, int size )
{   int i, sum;
    for( i=sum=0; i<size; i++ )
         sum += marks[i];
    return sum / size;
}
```

'Ragged' arrays are possible

eg: a ragged array of strings

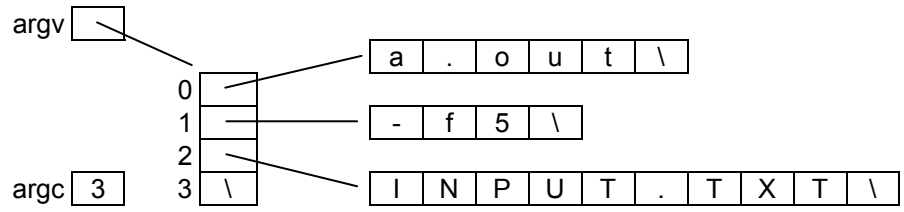


Command Line Arguments

Command line arguments are those words following the name of the executable on the UNIX command line.

EG: if the command line reads: **`a.out -f5 INPUT.TXT`**

the structures passed to the main function are:



To allow this the main function must be declared with argc & argv as arguments:

```
int main( int argc, char *argv[] )
```

Command line adder/subtractor

Based on a previous simple calculator program this adds or subtracts two numbers which are given on the command line. The operator +/- is also given on the command line.

EG: **`a.out + 12 7`**

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

int main( int argc, char *argv[] )
{
    int num1, num2, result;
    char op;

    if (argc != 4)
    {   printf( "bad command use" );
        return 1;
    }
    if (strlen( argv[1] ) != 1)
    {   printf( "\nunrecognised operator \"%s\"\n", argv[1] );
        return 1;
    }
    op = argv[1][0];
    num1 = atoi( argv[2] );
    num2 = atoi( argv[3] );
    if (op == '+')
    {   result = num1 + num2;
        printf( "\n %i + %i = %i\n", num1, num2, result );
    }
    else if (op == '-')
    {   result = num1 - num2;
        printf( "\n %i - %i = %i\n", num1, num2, result );
    }
    else
        printf( "\nunknown operator \"%c\"\n", op );

    return 0;
}
```