First C program, add 2 numbers

A program to read two numbers, add them & print result.

Algorithm

```
input 1st number → num1
input 2nd number → num2
num1 + num2 → result
output result
```

I/O Sample

enter two numbers to be added: 12 13

Sum is 25

Variables

name	range	type	description
num1	numeric	int	1st number for calculation
num2	numeric	int	2nd number for calculation
result	numeric	int	result

The Program

```
/*
 *  First C program, add 2 numbers
 */

#include <stdio.h>

int main()
{
    int num1, num2, result;                    /* declare variables */
    printf( "enter two numbers to be added: " );
    scanf( "%i%i", &num1, &num2 );             /* read 2 numbers   */
    result = num1 + num2;                      /* calculate result */
    printf( "\nSum is %i \n", result );        /* print result     */
    return 0;
}
```

A Simple Calculator: add or subtract two numbers

A program which reads two numbers then '+' or '-', adds or subtracts the numbers & prints the result.

Algorithm

```
input 1st number → num1
input 2nd number → num2
input '+' or '-'  → op
if op = '+' then
    num1 + num2 → result
    output result
else if op = '-' then
    num1 - num2 → result
    output result
else
    output message
```

I/O Sample

```
Enter two numbers then '+' or '-': 17 5 +

 17 + 5 = 22
```

Variables

name	range	type	description
num1	numeric	int	1st number
num2	numeric	int	2nd number
op	+ - {or error}	char	operator (plus, minus)
result	numeric	int	result of calculation

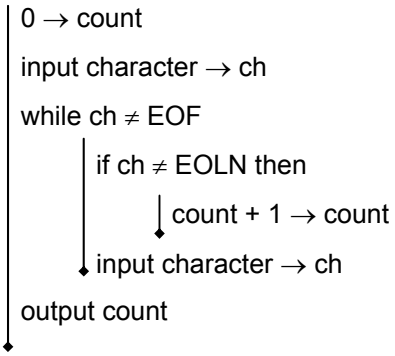
The Program

```
/*
 *  A simple calculator: add or subtract 2 numbers
 */

#include <stdio.h>

int main()
{
    int  num1, num2, result;
    char op;

    printf( "\nEnter two numbers then '+' or '-': " );
    scanf( "%i%i%c", &num1, &num2, &op );

    if (op == '+')              /* add numbers */
    {   result = num1 + num2;
        printf( "\n %i + %i = %i\n", num1, num2, result );
    }
    else if (op == '-')         /* subtract numbers */
    {   result = num1 - num2;
        printf( "\n %i - %i = %i\n", num1, num2, result );
    }
    else                        /* print message */
        printf( "\nunknown operator '%c'\n", op );

    return 0;
}
```

Count Characters In stdin

Reads & counts characters (except Newline) from stdin terminating at EOF.

Algorithm



I/O

For testing purposes only.
Input: a stream of characters ending in EOF.
Output: progress messages & count.

Variables

name	range	type	description
count	+ve integer	int	No. of chars counted
ch	character	char	character from input stream

Program version 1

```
/*
 * Count number of chars in stdin (except \n), terminates with EOF
 */

#include <stdio.h>

int main()
{   int  count;     // No. of chars counted
    char ch;        // char from stdin

    count = 0;
    printf( "Reading input..." );
    ch = getchar();
    while (ch != EOF)
    {   if (ch != '\n')
            count = count + 1;
        ch = getchar();
    }
    printf( "Finished\n%i characters read\n", count );
    return 0;
}
```

Program version 2

```
/*
 * Count number of chars in stdin (except \n), terminates with EOF
 * using embedded assignment & increment operator
 */

#include <stdio.h>

int main()
{   int  count;        // No. of chars counted
    char ch;           // char from stdin

    count = 0;
    printf( "Reading input..." );
    while ( (ch = getchar()) != EOF)
        if (ch != '\n')
            count++ ;
    printf( "Finished\n%i characters read\n", count );
    return 0;
}
```

Program version 3

```
/*
 * Count number of chars in stdin (except \n), terminates with EOF
 * using a for loop
 */

#include <stdio.h>
int main()
{   int  count;        // No. of chars counted
    char ch;           // char from stdin

    printf( "Reading input..." );

    for( count = 0; (ch = getchar())!= EOF; )
        if (ch != '\n')
            count++;

    printf( "Finished\n%i characters read\n", count );
    return 0;
}
```

Count Words in stdin

Count the number of words in stdin, terminating with EOF

Some Analysis

- Words are separated by 1 or more white-space chars.
- Need to recognise difference between printing chars & white-space chars - use isspace (predefined in ctype.h).
- Mechanism to distinguish between 1st space char after a word & subsequent spaces - use boolean variable 'word'.
- Variable 'word': set to false when reading spaces, true when reading printing characters.

Circumstance	value of 'word'	action
read 1st char of a new word	false → true	increment No. words counter
read char within a word	true	none
read 1st space char after word	true → false	none
read char within block of spaces	false	none

Algorithm

```
false → word
0 → count
while (input character → ch) ≠ EOF
    if word and isspace(ch) then
        false → word
    elseif not word and not isspace(ch) then
        true → word
        count ++
output count
```

I/O

For testing purposes only.
Input: a stream of characters ending in EOF.
Output: progress messages & count.

Variables

name	range	type	description
word	true/false	int	read status (see above)
count	+ve integer	int	No. of chars counted
ch	character	char	character from input stream

The Program version 1

```
/*
 * count the number of words in input stream
 */

#include <stdio.h>
#include <ctype.h>

int main()
{   int  word;      /* boolean, true if reading alphabetics */
    int  count;     /* word count  */
    char ch;        /* a character */

    word = count = 0;
    printf( "Reading input..." );

    while ((ch = getchar()) != EOF)
        if (word && isspace(ch))
            word = 0;
        else if (!word && !isspace(ch))
        {   word = 1;
            count++;
        }

    printf( "Finished\nNo. of words = %i\n", count );
    return 0;
}
```

The Program version 2 (without using ctype.h & isspace)

```
/*
 * count the number of words in input stream
 */

#include <stdio.h>

int isaspace( char );        /* prototype for isaspace */

int main()
{   int  word;      /* boolean, true if reading alphabetics */
    int  count;     /* word count  */
    char ch;        /* a character */

    word = count = 0;
    printf( "Reading input..." );

    while ((ch = getchar()) != EOF)
        if (word && isaspace(ch))
            word = 0;
        else if (!word && !isaspace(ch))
        {   word = 1;
            count++;
        }

    printf( "Finished\nNo. of words = %i\n", count );
    return 0;
}

int isaspace( char c )
{    return c == ' ' || c == '\n' || c == '\t';
}
```