Range of different data types in C

Suppose you need to store an integer value which can range from zero to one million. Which is the smallest type you can use? There is no general rule; it depends on the C compiler and target machine. You can use the 'MIN' and 'MAX' macros in `limits.h` to determine which type will work.

Each signed integer type has a pair of macros which give the smallest and largest values that it can hold. Each unsigned integer type has one such macro, for the maximum value; the minimum value is, of course, zero.

The values of these macros are all integer constant expressions. The 'MAX' and 'MIN' macros for `char` and `short int` types have values of type `int`. The 'MAX' and 'MIN' macros for the other types have values of the same type described by the macro—thus, `ULONG_MAX` has type `unsigned long int`.

`SCHAR_MIN`

> This is the minimum value that can be represented by a `signed char`.

`SCHAR_MAX`

`UCHAR_MAX`

> These are the maximum values that can be represented by a `signed char` and `unsigned char`, respectively.

`CHAR_MIN`

> This is the minimum value that can be represented by a `char`. It's equal to `SCHAR_MIN` if `char` is signed, or zero otherwise.

`CHAR_MAX`

> This is the maximum value that can be represented by a `char`. It's equal to `SCHAR_MAX` if `char` is signed, or `UCHAR_MAX` otherwise.

`SHRT_MIN`

> This is the minimum value that can be represented by a `signed short int`. On most machines that the GNU C Library runs on, `short` integers are 16-bit quantities.

`SHRT_MAX`

`USHRT_MAX`

> These are the maximum values that can be represented by a `signed short int` and `unsigned short int`, respectively.

`INT_MIN`

> This is the minimum value that can be represented by a `signed int`. On most machines that the GNU C Library runs on, an `int` is a 32-bit quantity.

`INT_MAX`

`UINT_MAX`

These are the maximum values that can be represented by, respectively, the type `signed int` and the type `unsigned int`.

`LONG_MIN`

This is the minimum value that can be represented by a `signed long int`. On most machines that the GNU C Library runs on, `long` integers are 32-bit quantities, the same size as `int`.

`LONG_MAX`

`ULONG_MAX`

These are the maximum values that can be represented by a `signed long int` and `unsigned long int`, respectively.

`LLONG_MIN`

This is the minimum value that can be represented by a `signed long long int`. On most machines that the GNU C Library runs on, `long long` integers are 64-bit quantities.

`LLONG_MAX`

`ULLONG_MAX`

These are the maximum values that can be represented by a `signed long long int` and `unsigned long long int`, respectively.

`LONG_LONG_MIN`

`LONG_LONG_MAX`

`ULONG_LONG_MAX`

These are obsolete names for `LLONG_MIN`, `LLONG_MAX`, and `ULLONG_MAX`. They are only available if `_GNU_SOURCE` is defined (see Feature Test Macros). In GCC versions prior to 3.0, these were the only names available.

`WCHAR_MAX`

This is the maximum value that can be represented by a `wchar_t`

The header file `limits.h` also defines some additional constants that parameterize various operating system and file system limits

//Program to find the range of different data types in C:

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <float.h>
int main(void) {
   printf("CHAR_BIT    :  %d\n", CHAR_BIT);
   printf("CHAR_MAX    :  %d\n", CHAR_MAX);
   printf("CHAR_MIN    :  %d\n", CHAR_MIN);
   printf("INT_MAX     :  %d\n", INT_MAX);
   printf("INT_MIN     :  %d\n", INT_MIN);
```

```
    printf("LONG_MAX    :  %ld\n", (long) LONG_MAX);
    printf("LONG_MIN    :  %ld\n", (long) LONG_MIN);
    printf("SCHAR_MAX   :  %d\n", SCHAR_MAX);
    printf("SCHAR_MIN   :  %d\n", SCHAR_MIN);
    printf("SHRT_MAX    :  %d\n", SHRT_MAX);
    printf("SHRT_MIN    :  %d\n", SHRT_MIN);
    printf("UCHAR_MAX   :  %d\n", UCHAR_MAX);
    printf("UINT_MAX    :  %u\n", (unsigned int) UINT_MAX);
    printf("ULONG_MAX   :  %lu\n", (unsigned long) ULONG_MAX);
    printf("USHRT_MAX   :  %d\n", (unsigned short) USHRT_MAX);
    printf("FLT_MAX     :  %g\n", (float) FLT_MAX);
    printf("FLT_MIN     :  %g\n", (float) FLT_MIN);
    printf("-FLT_MAX    :  %g\n", (float) -FLT_MAX);
    printf("-FLT_MIN    :  %g\n", (float) -FLT_MIN);
    printf("DBL_MAX     :  %g\n", (double) DBL_MAX);
    printf("DBL_MIN     :  %g\n", (double) DBL_MIN);
    printf("-DBL_MAX    :  %g\n", (double) -DBL_MAX);
    return (0);
}
```

Analysis for auto increment and decrement operator:

Int x=2, y;
y= ++x + ++x + ++x;
printf("\n%d %d",y,x); // o/p 13,5
decomposing the calculation:
Y= (x=1+x) + (x=1+x) + ++x
= (x=3)+(x=3+1=4)+ ++x
= (x=4) + (x=4) + ++x
= 8+ (x=1+x)
= 8+ (x=1+4=5)
= 8+5=13 (y=13,x=5)
	Int x=2, y;
	y= ++x + x++ + x--;
	=(x=1+x=1+2=3) + (x=(x+1) + x--
	= (x=3)+x=(3)+1 + x--
	=6+ (x=x-1) +1
	=6+(x=3)// +1 -1
	=9 (y=9,x=3)

Int x=2, y;
y= ++x + x++ + --x + x--;
=(x=1+x) + (x=x+1) + --x + x--;
= (x=3)+(x=3 //+1) + --x + x--;
= 6 + (x=-1+x=-1+4) + x-- //+1
=(6+x=2)=8 + (x=2//-1) //+1
= 8 + 2 //+1-1
=10 (y=10,x=2)
	Int x=2,y;
	y=++x + x-- + ++x + x++;
	=(x=1+x) + (x=x-1) + ++x + x++
	= (x=3)+(x=3)+ ++x + x++ //-1
	=6+(x=1+x=4) + (x=x+1)
	=6+4 + (x=4) //-1+1
	=10+4 //-1+1
	=14 (y=14,x=4)

Int x=2,y;
x=++x + x-- + ++x ;
= (x=1+x) + (x=x-1) + ++x
= (x=3)+(x=3) + ++x //-1
=6+(x=1+x) //-1
=6+x=4 //-1
	Int x=2,y
	x=++x + x++ + x++ + --x;
	= (x=1+x) + (x=x+1) + x++ + --x
	= (x=3) + (x=3//+1) + (x=x+1) + --x
	= (3+3=6) + (x=3//+1) + --x //+1
	=(6+3=9) + (x=-1+x) //+1+1

=10 -1 =9 (x=9)	=9+(x=-1+3=2) //+2
	=(x= 11) //+2
	=13 (x=13)

Int x=2,y
y=++x + x++ + x++ + --x;
= (x=1+x) + (x=x+1) + x++ + --x
= (x=3) + (x=3//+1) + (x=x+1) + --x
= (3+3=6) + (x=3//+1) + --x //+1
=(6+3=9) + (x=-1+x) //+1+1
=9+(x=-1+3=2) //+2
=(y= 11) //+2
(y=11,x=4)
	int x=2, y;
	y = x++ + ++-x + x-- + -x ;
	= (x=x+1) + (x=1+x) + x-- + -x
	= (x=2) + (x=3) + (x=x-1) + -x //+1
	= (3+3=6) + (x=3) + -x //+1 -1
	=9 + -3 //+1-1
	Y=6, x=3