```c
#include <stdlib.h>

// static storage
float a[200];

int main(void)
{
    // stack storage allocation, "automatic" variables
    float b[300];

    // heap storage
    float * c = (float *)malloc(400*sizeof(float));
    *(c+45) = 93.4;
    c[45] = 93.4;

    free(c);
    return 0;
}
```

```c
#include<stdio.h>

enum Rainbow {Red,Orange,Yellow,Green,Blue,Indigo,Violet};


int main(void)
{
    printf("Hello world\n");
    enum Rainbow bow;
    bow = Yellow;

    if( bow == Yellow )
    {
        printf("Rainbow is Yellow\n");
    }
    bow = Indigo;
    bow = 200;
    switch( bow )
    {
        case Red:
            printf("Red %d\n",bow);
            break;
        case Orange:
            printf("Orange %d\n",bow);
            break;
        case Yellow:
            printf("Yellow %d\n",bow);
            break;
        case Green:
            printf("Green %d\n",bow);
            break;
        case Blue:
            printf("Blue %d\n",bow);
            break;
        case Indigo:
            printf("Indigo %d\n",bow);
            break;
        case Violet:
            printf("Violet %d\n",bow);
            break;
    }
    bow = 200;

    printf("Bytes used to store enum Rainbow = %d\n",sizeof(enum Rainbow));
}
```

```c
#include <stdio.h>

/* Function prototypes. */
void take_float( float );
void take_int( int );
void take_int2( int * );
void take_array( int *, int );

int main( void )
{
    float a = 57.4;
    printf("\nfloat a has value %f at address %p\n",a,&a);
    take_float(a);

    int b = 33;
    printf("\nint b has value %d at address %p\n",b,&b);
    take_int(b);

    int c = 99;
    printf("\nint c has value %d at address %p\n",c,&c);
    take_int2(&c);

    int d[40];
    printf("\nint d[40] starts at address %p, i.e. %p\n",d,&d[0]);
    take_array(d,40);

    printf("\n");
}

/* Function implementations. */
void take_float( float x)
{
    printf("Inside function take_float, received %f at address %p\n",x,&x);
}

void take_int( int x )
{
    printf("Inside function take_int, received %d at address %p\n",x,&x);
}

void take_int2( int * x )
{
    printf("Inside function take_int2, received %d at address %p\n",*x,x);
}

void take_array( int * x, int length )
{
    printf("Inside take_array, received array at address %p\n",x);
}
```

```c
#include<stdio.h>

struct piece
{
    int a;
    float b;
};

void func1( struct piece x )
{
    x.a = 99;
    x.b = 99.9f;
}

void func2( struct piece * x )
{
    x->a = 99;
    x->b = 99.9f;
}

int main(void)
{
    struct piece i = {2,4.5f};

    printf("i.a = %i, i.b = %f",i.a, i.b);

    printf("\nCalling func1");
    func1(i);
    printf("\nAfter func1:");
    printf("\n\ti.a = %i, i.b = %f",i.a, i.b);

    printf("\nCalling func2");
    func2(&i);
    printf("\nAfter func2:");
    printf("\n\ti.a = %i, i.b = %f",i.a, i.b);
    printf("\n");
}
```

```c
/* Including other files */

#include <stdio.h>
#include "linked_list.h"

/* Conditional compilation */

// For GCC, predefined macros: http://gcc.gnu.org/onlinedocs/cpp/Predefined-Macros.html
#ifdef _WIN64
    // definitions specific to 64 bit windows
#elif _WIN32
    //...
#elif __APPLE__
    // for Macs
#elif __posix
    // for unix
#endif

/* Macros */

#define PI      3.141592653589793
#define TWOPI   6.283185307179586
#define TORADIANS(x)    ((x) * 0.017453292519943295)


int main( int argc, char * argv[] )
{
    printf("Hello World!\n");
    return 0;
}
```

```c
/* Including other files */

#include <stdio.h>
#include "linked_list.h"

/* Conditional compilation */

// For GCC, predefined macros: http://gcc.gnu.org/onlinedocs/cpp/Predefined-Macros.html
#ifdef _WIN64
    // definitions specific to 64 bit windows
#elif _WIN32
    //...
#elif __APPLE__
    // for Macs
#elif __posix
    // for unix
#endif

/* Macros */

#define PI      3.141592653589793
#define TWOPI   6.283185307179586
#define TORADIANS(x)    ((x) * 0.017453292519943295)


int main( int argc, char * argv[] )
{
    printf("Hello World!\n");
    return 0;
}
```

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

/* Generate all permutations of a set. Knuth, The Art of Computer
Programming, Volume 4, Fascicle 2, Section 7.2.1.2  Original algorithm
due to Trotter CACM 1962, and Johnson.

-Scot Morse
*/

#define NMAX 12

// What to do with each permutation
void visit(int P[], int N)
{
    int i;
    for( i = 1; i <= N; ++i )
    {
        printf("%i ",P[i]);
    }
    printf("\n");
}

int main( void )
{
    printf("N = %i\n",NMAX);
    int a[NMAX+2];      // a_1 through a_n
    int c[NMAX+2];      // c_1 through c_n
    int o[NMAX+2];      // o_1 through o_n

    int n = NMAX;
    int s,q,temp;
    // Initialize
    int j;
    for( j = 1; j <= n; ++j )
    {
        a[j] = j;
        c[j] = 0;
        o[j] = 1;
    }
P2:     // Visit a[1:n]
//  visit(a,n);
    // Prepare for change
P3: j = n;
    s = 0;
P4: // Ready to change?
    q = c[j] + o[j];
    if( q < 0 )
        goto P7;
    else if( q == j )
        goto P6;
P5: // Change (exchange)
    temp = a[j-c[j]+s];
    a[j-c[j]+s] = a[j-q+s];
    a[j-q+s] = temp;
    c[j] = q;
    goto P2;
P6: // Increase s
    if( j ==1 )
        goto END;
    else
        ++s;
P7: // Switch direction
    o[j] = -o[j];
```

```
    --j;
    goto P4;

END:
    printf("\nDone\n");
}
```

```c
#include <stdio.h>
#include <math.h>

double fun( int a, double b )
{
    return a*2.0+b;
}

struct Complex
{
    float r;
    float i;
};

int main( void )
{
    int * pi = 0x0045;
//  *pi = 43;
    int A = 999999;
    int * p = &A;
    float * pF = (float *)&A;
    printf("pF = %p, *pF = %f\n",pF,*pF);
//  int & r = A;                        // C++ only

    printf("A = %i\n",A);
    printf("&A = %i\n",&A);
    printf("&A = %p\n",&A);
    printf("p = %p\n",p);
    printf("&p = %p\n",&p);
    printf("*p = %i\n",*p);
//  printf("r = %i\n",r);       // C++ only

    *p = 22;
    printf("A = %i\n\n",A);

//  r = 99;                     // C++ only
//  printf("A = %i\n",A);

    struct Complex z = {1.0,0.5};
    struct Complex * pZ = &z;
    z.r = 2.0;
    pZ->i = 5.0;
    printf("z = (%f,%f)\n\n",pZ->r,pZ->i);

    (*pZ).i = 99;
    printf("z = (%f,%f)\n\n",pZ->r,pZ->i);


    // Pointer to array
    int arr[100];
    int i;
    for(i = 0; i < 99; ++i)
        arr[i] = 4*i;
    int * ptr = arr;            // or int * ptr = &arr[0];
    for(i = 0; i < 99; ++i)
        *(arr+i) = 40*i;
    printf("arr[1] = %i\n\n",arr[1]);


    // Pointer to a function
    double (*fp)(int,double);
    fp = fun;
    double answer = (*fp)(2,10.0);
    printf("answer = %f\n",answer);
```

```
    answer = fp(2,10.0);
    printf("answer = %f\n",answer);

    // Powerful but dangerous stuff
    int val = 15;
    printf("val is %i or %X\n",val,val);
    unsigned char * pChar = (unsigned char *)&val;
    printf("Bytes of val: %i, %i, %i, %i\n",*pChar,*(pChar+1),
           *(pChar+2),*(pChar+3));
    // Intel is little endian LSB is to the left
    *(pChar+3) = 0xff;
    printf("val was 15, now is %i\n\n",val);

    return 0;
}
```

```c
#include <stdio.h>
#include <stddef.h>
#include <limits.h>
#include <inttypes.h>

int main(void)
{
    /* Data types. YMMV depending on hardware. */
    printf("Sizes of various data types, in bytes.\n");
    printf("Type\t\tbytes\n");
    printf("WORD_BIT\t%d:\n",WORD_BIT);
    printf("char\t\t%2lu \n", sizeof(char));
    printf("short\t\t%2lu \n", sizeof(short));
    printf("int\t\t%2lu \n", sizeof(int));
    printf("float\t\t%2lu \n", sizeof(float));
    printf("double\t\t%2lu \n", sizeof(double));
    printf("size_t\t\t%2lu \n", sizeof(size_t));
    printf("wchar_t\t\t%2lu \n", sizeof(wchar_t));
    printf("ptrdiff_t\t%2lu \n", sizeof(ptrdiff_t));
    printf("---\n");
    printf("int8_t\t\t%2lu \n", sizeof(int8_t));
    printf("uint8_t\t\t%2lu \n", sizeof(uint8_t));
    printf("int16_t\t\t%2lu \n", sizeof(int16_t));
    printf("uint16_t\t%2lu \n", sizeof(uint16_t));
    printf("int32_t\t\t%2lu \n", sizeof(int32_t));
    printf("uint32_t\t%2lu \n", sizeof(uint32_t));
    printf("int64_t\t\t%2lu \n", sizeof(int64_t));
    printf("uint64_t\t%2lu \n", sizeof(uint64_t));
    printf("---\n");
    printf("unsigned short int\t%2lu \n", sizeof(unsigned short int));
    printf("long long\t\t%2lu \n", sizeof(long long));
    printf("long double\t\t%2lu \n", sizeof(long double));
    printf("---\n");
    printf("char *\t\t%2lu \n", sizeof(char *));
    printf("short *\t\t%2lu \n", sizeof(short *));
    printf("int *\t\t%2lu \n", sizeof(int *));
    printf("float *\t\t%2lu \n", sizeof(float *));
    printf("double *\t%2lu \n", sizeof(double *));
    printf("void *\t\t%2lu \n", sizeof(void *));
    printf("char**\t\t%2lu \n", sizeof(char**));
    int a = 5;
    printf("a = 5\n");
    printf("&a\t\t%2lu \n", sizeof(&a));
}
```

```c
#include <stdio.h>

struct Stuff
{
    int a;
    unsigned char b;
    double c[300];
};

int main(void)
{
    struct Stuff myStuff;
    printf("myStuff.b = %x\n",(unsigned int)(myStuff.b));
    myStuff.a = 54;
    myStuff.b = 'b';
    myStuff.c[45]= 67.5;
    printf("sizeof(struct Stuff) = %d\n",sizeof(struct Stuff));
    printf("sizeof(unsigned char) = %d\n",sizeof(unsigned char));
    printf("sizeof(int) = %d\n",sizeof(int));
    printf("sizeof(double) = %d\n",sizeof(double));

    printf("myStuff.b = %u\n",myStuff.b);
    printf("myStuff.c[45] = %d\n",myStuff.c[45]);

}
```

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



// structs

struct Book
{
    char author[64];
    char title[128];
    int id;
    unsigned int numberOfPages;
};

void doSomething(struct Book * pb)
{
    printf("author is %s\n",pb->author);
    pb->numberOfPages = 5;
}

typedef struct
{
    int x;
    int y;
}  Point;

// union

union TripleConstraint
{
    long int cost;
    double quality;
    char speed;
};

typedef union
{
    char kph;
    int mph;
} Speed;

int main( int argc, char * argv[] )
{
    struct Book myBook = {"Jules Verne","20,000 Leagues Under the Sea",45938223,324};
    doSomething( &myBook );
    printf("number of pages is %u\n",myBook.numberOfPages);
    printf("My book is %s by %s.\n",myBook.title,myBook.author);
    printf("sizeof(myBook) = %zd bytes\n",sizeof(myBook));
    printf("sizeof(struct Book) = %zd bytes\n\n",sizeof(struct Book));

    Point p1 = {100,200};
    Point p2;
    p2.x = 5;
    p2.y = 20;
    printf("p1 is at %d,%d while p2 is at %d,%d.\n",p1.x,p1.y,p2.x,p2.y);
    printf("sizeof(p1) = %zu bytes\n",sizeof(p1));
    printf("sizeof(Point) = %zu bytes\n",sizeof(Point));
    printf("sizeof(int) = %zu bytes\n",sizeof(int));
    printf("sizeof(unsigned int) = %zu bytes\n",sizeof(unsigned int));
    printf("sizeof(char) = %zu bytes\n\n",sizeof(char));

    union TripleConstraint t;
    t.cost = 45002032;
```

```
    printf("sizeof(t) = %zu bytes\n",sizeof(t));
    printf("sizeof(union TripleConstraint) = %zu bytes\n",sizeof(union TripleConstraint
));
    printf("sizeof(long int) = %zu bytes\n\n",sizeof(long int));

    Speed sp;
    sp.kph = 4;
    printf("sizeof(sp) = %zu bytes\n",sizeof(sp));

    exit(EXIT_SUCCESS);
}
```

```c
#include <stdio.h>

/* Declare a union type.  Unions are like alternation,
    i.e. OR, whereas structs are like composition, i.e
    AND.*/

union Multi
{
    int id;
    float percentage;
    unsigned char pieces[4];
};

/* Helper function to print out the contents of a
    union Multi.*/
void printMulti( union Multi u )
{
    printf("int\t\tfloat\t\tchar[0]\t[1]\t[2]\t[3]\thex\n");
    printf("%11d\t%g\t%d\t%d\t%d\t%d\t0x%08X\n",u.id,u.percentage,
            u.pieces[0],u.pieces[1],u.pieces[2],u.pieces[3],*((unsigned int*)&u));
    printf("\n");
}

/* Entry function. */
int main(void)
{
    // Create (allocate memory) for a union Multi
    union Multi data;
    printf("\nsizeof(union Multi) = %d\n\n",sizeof(union Multi));

    // What's in it now?
    printMulti( data );

    // Write to the int
    data.id = 65536;
    printf("_____data.id = 65536_____\n");
    printMulti( data );

    data.percentage = 91.456e12;
    printf("_____data.percentage = 91.456e12_____\n");
    printMulti( data );

    data.pieces[0] = 0x01;
    data.pieces[1] = 0x00;
    data.pieces[2] = 0xff;
    data.pieces[3] = 0xaa;
    printf("_____data.pieces = 0x0100ffaa_____\n");
    printMulti( data );

}
```

```c
#include <stdio.h>
#include <string.h>

int add(int a, int b)
{
    return a + b;
}

int main(void)
{
    int a = 5;
    int b = 7;
    int c = add(a,b);
    printf("a + b = %d + %d = %#010x + %#010x = %#010x\n",a,b,a,b,c);
    float x = 5.5;
    float d = add(a,x);
    unsigned int ix;
    memcpy(&ix, &x, sizeof(ix));
    unsigned int id;
    memcpy(&id, &d, sizeof(id));
    printf("a + x = %d + %f = %#010x + %#010x = %#010x = %f\n",a,x,a,ix,id,d);
}
```