May 02, 04 16:38 **eg01−hello.c** Page 1/1

```
/*
 * PURPOSE:
 *   Illusrate #include, int main(), printf, ’\n’, and return 0.
 */

#include <stdio.h>

int main()
{
    printf("hello world\n");
    return 0;
}
```

May 02, 04 16:40 **eg02−factorial.c** Page 1/1

```
/*
 * PURPOSE:
 *   Introduce functions definitions, and simple control structure.
 */

#include <stdio.h>

int fac(int n)
{
    if (n == 0)
        return 1;
    else
        return n*fac(n−1);
}

int main()
{
    int i;
    for (i = 0; i < 10; i++)
    {
        printf("%d! is %d\n", i, fac(i));
    }
    return 0;
}
```

May 02, 04 16:41 **eg03−boolean.c** Page 1/1

```
/*
 * PURPOSE:
 *   Show that in C, non−zero values are taken to be true
 *   and zero is take to be false.
 */

#include <stdio.h>

int main()
{
    int count = 10;
    while (count)
    {
        printf("%d\n", count);
        count−−;
    }
}
```

May 02, 04 16:43 **eg04−assign.c** Page 1/1

```
/*
 * PURPOSE:
 *   Show that assignment statement in C returns the
 *   assigned value.  Which can lead to very compact
 *   but unreadable code.
 */

#include <stdio.h>

int main()
{
        int i, j;
        printf("sum is %d\n", (i = 4) + (j = 9));
        printf("i is %d\nj is %d\n", i, j);
        return 0;
}
```

May 02, 04 16:44 **eg05−goto.c** Page 1/1

```
/*
 * PURPOSE:
 *   C has a command call goto which allows execution to jump to
 *   specified location.  This can lead to unreadable code if used
 *   unnecessarily (loops and branch is still prefered over goto).
 */
#include <stdio.h>

int main()
{
    int i = 10;
loop:
    if (i > 0) {
            printf("%d\n", i);
            i−−;
            goto loop;
    }
}
```

May 02, 04 16:46 **eg06−memory.c** Page 1/1

```
/*
 * PURPOSE:
 *   Illustrate the "*" and "&" operator.
 */

#include <stdio.h>

int main()
{
    int x = 1;
    int *y;

    y = &x;
    printf("&x is %x\n", &x);
    printf("*y is %d\n", *y);
    printf("*&x is %d\n", *&x);

    *y = 4;
    printf("after *y = 4, x is %d\n", x);

    printf("now, try y = 4 and print *y\n");
    y = 4;
    printf("after y = 4, *y is %d\n", *y);

    return 0;
}
```

May 02, 04 17:13 **eg07−malloc.c** Page 1/1

```
/*
 * PURPOSE:
 *   Show that a declared pointer points to undefined
 *   location.  We need to allocate appropriate amount
 *   of memory for pointer before using it.
 */
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int *y;
    y = malloc(sizeof(int));
    *y = 4;

    return 0;
}
```

May 02, 04 16:52 **eg08−addptr.c** Page 1/1

```
#include <stdio.h>
/*
 * PURPOSE:
 *
 *  Introduces pointer arithmetics, and local variables.
 *  Careless use of pointer arithmetics can lead to buggy programs!
 */

int main()
{
    int *p;
    int x = 1;
    int y = 6754378;

    p = &x;
    *(p−1) = 12345;
    printf("x is %d and y is %d\n", x, y);

    return 0;
}
```

May 01, 04 22:32 **eg09−global.c** Page 1/1

```c
#include <stdio.h>
/*
 * PURPOSE:
 *
 *  Introduces global variables.  Notice that the address range
 *  is different, and the address of x is smaller than y.
 *
 */

int x = 1;
int y = 6754378;

int main()
{
    int *p;

    p = &x;
    printf("&x is %p and &y is %p\n", &x, &y);

    return 0;
}
```

May 02, 04 17:23 **eg10−type.c** Page 1/1

```c
/*
 * PURPOSE:
 *   Illustrate that different types can treat
 *   the same number differently.
 */
#include <stdio.h>

int main()
{
    int ix = −190;
    unsigned int uix = ix;
    char c = ix;
    unsigned char uc = ix;
    float f = ix;
    double d = ix;

    printf("%u\n", uix);
    printf("%c\n", c);
    printf("%c\n", uc);
    printf("%f\n", f);
    printf("%f\n", d);

    return 0;
}
```

May 02, 04 17:23 **eg11−sizeof.c** Page 1/1

```
/*
 * PURPOSE:
 *   Introduces various variable type and their size
 *   (platform dependent).
 */
#include <stdio.h>

int main()
{
    printf("sizeof(long long) is %d\n", sizeof(long long));
    printf("sizeof(long) is %d\n", sizeof(long));
    printf("sizeof(int) is %d\n", sizeof(int));
    printf("sizeof(short) is %d\n", sizeof(short));
    printf("sizeof(char) is %d\n", sizeof(char));
    printf("sizeof(float) is %d\n", sizeof(float));
    printf("sizeof(double) is %d\n", sizeof(double));

    return 0;
}
```

May 02, 04 17:14 **eg12−ptrtype.c** Page 1/1

```
/*
 * PURPOSE:
 *   Show that different pointer type interpret the same bits
 *   differently.
 */
#include <stdio.h>

int main()
{
    int ix = 65;
    unsigned int *ui = &ix;
    char *c = &ix;
    unsigned char *uc = &ix;
    float *f = &ix;
    double *d = &ix;

    printf("*ui %u\n", *ui);
    printf("*c is %c\n", *c);
    printf("*(c+1) is %c\n", *(c+1));
    printf("*(c+2) is %c\n", *(c+2));
    printf("*(c+3) is %c\n", *(c+3));
    printf("*uc is %c\n", *uc);
    printf("*f is %e\n", *f);
    printf("*d %e\n", *d);

    return 0;
}
```

```
/*
 * PURPOSE:
 *   Same as previous example except that [] notation is
 *   use for pointer arithmetic.
 */
#include <stdio.h>

int main()
{
    int ix = 65;
    unsigned int *ui = &ix;
    char *c = (char *)&ix;
    unsigned char *uc = (unsigned char *)&ix;
    float *f = (float *)&ix;
    double *d = (double *)&ix;

    printf("*ui %u\n", *ui);
    printf("*c is %c\n", c[0]);
    printf("*(c+1) is %c\n", c[1]);
    printf("*(c+2) is %c\n", c[2]);
    printf("*(c+3) is %c\n", c[3]);
    printf("*uc is %c\n", *uc);
    printf("*f is %e\n", *f);
    printf("*d %e\n", *d);

    return 0;
}
```

```
#include <stdio.h>
/*
 * PURPOSE:
 *   Show that negative index is allowed in C.
 */

int main()
{
    int x = 1;
    int y = 6754378;
    int *p;

    p = &x;
    printf("p is %p and p−1 is %p\n", p, p−1);
    printf("*p is %d\n", p[0]);
    printf("*(p−1) is %d\n", p[−1]);

    *(p−1) = 123456;
    printf("y is %d\n", y);

    return 0;
}
```

```
/*
 * PURPOSE:
 *   Show various ways of declaring an array.
 */
#include <stdio.h>

int main()
{
    int count[10];

    double d[2] = {1.0, 2.0, 4.0};
    int x[] = {65, 65, 65, 65, 65};
    char str[5] = {'h', 'e', 'l', 'l', 'o'};
    short s[5] = {1000,1000,1000};

    printf("%s\n", str);

    return 0;
}
```

```
/*
 * PURPOSE:
 *   Introduces string -- which is an array of char terminated by 0.
 */
#include <stdio.h>

int main()
{
    char str1[6] = {'h', 'e', 'l', 'l', 'o', 0};
    char str2[6] = "hello";
    char *str3 = "hello";
    char *str4;
    char *str5;

    str4 = "hello";
    str5 = str4;
}
```

```c
/*
 * PURPOSE:
 *  Introduce some predefined function to manipulate stirngs.
 */
#include <stdio.h>
#include <string.h>
int main()
{
        char s[12];
        char t[24];
        strncpy(s, "UNIX is fun", 12);
        printf("%d\n", strlen(s));
        printf("%d\n", strncmp(s, "Unix is fun", 12));

        strncpy(t, "UNIX is fun", 12);
        strncat(t, ", or is it?", 13);
        printf("%s\n", t);

        return 0;
}
```

```c
/*
 * PURPOSE:
 *   Show how to pass command line arguments to C
 *   program using array of strings.
 */
#include <stdio.h>

int
main(int argc, char *argv[])
{
        int i;
        printf("calling with %d arguments\n", argc);
        for (i = 0; i < argc; i++)
        {
                printf("argument %d is %s\n", i, argv[i]);
        }
        return 0;
}
```