Sort_integer.c

```
/*
sort_integer.c -- shows simple integer sort using qsort
*/
#include <stdlib.h>
#include <stdio.h>

#define MAX 100

int intcomparison(const void *v1, const void *v2);

int main(void)
{
        int i, arr[MAX], arrsort[MAX];
/* Get random integers from the rand() function */
        srand(17);
        printf("RAND_MAX=%10d\n", RAND_MAX);            /* check operating
system limit value */
        for(i=0;i<MAX;i++)
        {
                arr[i] = rand(); /* fill arr[,] with random integers*/
                arrsort[i] = arr[i];
        }
        qsort(arrsort, MAX, sizeof(arrsort[0]), intcomparison);
        for(i=0;i<MAX;i++)
        {
                printf("%10d  arr[%10d]  arrsort[%10d]\n", i, arr[i],
arrsort[i]);
        }
        printf("\n\n");
        return(0);
}
int intcomparison(const void *v1, const void *v2)
{
    return (*(int *)v1 - *(int *)v2);
}
```

Sort_integer_2.c

```
/*
sort_integer.c -- shows simple integer sort using qsort
*/
#include <stdlib.h>
#include <stdio.h>

#define MAX 100

int intcomparison(const void *v1, const void *v2);

int main(void)
{
        int i, arr[MAX], arrsort[MAX];
/* Get random integers from the rand() function */
        srand(17);
        printf("RAND_MAX=%10d\n", RAND_MAX);            /* check operating
system limit value */
        for(i=0;i<MAX;i++)
        {
                arr[i] = rand(); /* fill arr[,] with random integers*/
                arrsort[i] = arr[i];
        }
        qsort(arrsort, MAX, sizeof(arrsort[0]), intcomparison);
        for(i=0;i<MAX;i++)
        {
                printf("%10d  arr[%10d]  arrsort[%10d]\n", i, arr[i],
arrsort[i]);
        }
        printf("\n\n");
        return(0);
}
int intcomparison(const void *v1, const void *v2)
{
        const int *a1 = v1;
        const int *a2 = v2;
        return (*a1 - *a2);
}
```

Sort_double.c

```
/*
sort_double.c -- shows simple double sort using qsort
*/
#include <stdlib.h>
#include <stdio.h>

#define MAX 100

int doublecomparison(const void *v1, const void *v2);

int main(void)
{
        int i;
        double arr[MAX], arrsort[MAX];
/* Get random integers from the rand() function */
        srand(17);
        printf("RAND_MAX=%10d\n", RAND_MAX);            /* check operating
system limit value */
        for(i=0;i<MAX;i++)
        {
                arr[i] = rand()/((double)RAND_MAX + 1); /* fill arr[,] with
random doubles*/
                arrsort[i] = arr[i];
        }
        qsort(arrsort, MAX, sizeof(double), doublecomparison);
        for(i=0;i<MAX;i++)
        {
                printf("%10d  arr[%10.6f]  arrsort[%10.6f]\n", i, arr[i],
arrsort[i]);
        }
        printf("\n\n");
        return(0);
}
int doublecomparison(const void *v1, const void *v2)
{
        const double *a1 = v1;
        const double *a2 = v2;
        if(*a1 < * a2)return -1;
        else if(*a1 == *a2)return 0;
        else return 1;
}
```

Sort_structure.c

```
/*
sort_structure.c -- shows simple double sort using qsort
*/
#include <stdlib.h>
#include <stdio.h>

#define MAX 100

typedef struct {int icount;
               double data;
} kpsorter;

int structcomparison(const void *v1, const void *v2);
int real2_cmp(const void *s1, const void *s2);

int main(void)
{
        int i;
        double arr[MAX], arrsort[MAX];
/* Get random integers from the rand() function */
        kpsorter recordset[MAX];
        srand(17);
        printf("RAND_MAX=%10d\n", RAND_MAX);            /* check operating
system limit value */
        for(i=0;i<MAX;i++)
        {
                arr[i] = rand()/((double)RAND_MAX + 1); /* fill arr[,] with
random doubles*/
                arrsort[i]=arr[i];
                recordset[i].icount=i;
                recordset[i].data=arr[i];
        }
        qsort(recordset, MAX, sizeof(kpsorter), structcomparison);
        for(i=0;i<MAX;i++)
        {
                printf("%10d %10d  arr[%10.6f]  arrsort[%10.6f]\n", i,
recordset[i].icount, arrsort[i], recordset[i].data);
        }
        printf("\n\n");
        return(0);
}
int structcomparison(const void *v1, const void *v2)
{
        kpsorter *p1=(kpsorter *)v1;
        kpsorter *p2=(kpsorter *)v2;

        if(p1->data < p2->data)
                return -1;
        else if (p1->data == p2->data)
                return 0;
        else
                return 1;
}
```