Code:

```
// Headers for library functions
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>

// Forward declare
void Initalize();
void Finalize();
void Grow();
void Shrink();
void PrintVector();
void AddElement();
void RemoveElement();
void InsertElement();
int PrintMenu();
int Run();

// Global variables
int size;
int count;
double *v;

/**
* Initalize the global variables. Allocate some space for the vector.
*/
void Initalize()
{
    size = 2;
    count = 0;
    v = (double *) malloc(size * sizeof(double));

    // If allocation failed, print error
    if (v == NULL) {
        printf("ERROR: Out of memory, malloc failed in initalize\n");
        exit(0);
    }
}
```

```
/**
* Free the allocated memory
*/
void Finalize()
{
    free(v);
}
```

```
/**
* Grow the vector
*/
void Grow()
{
    printf("Vector grown\n");
    printf("Previous capacity: %d elements\n", size);

    // Double the size of the vector and allocate more space
    size *= 2;

    // Reallocate previous values and make the vector twice as big
    v = realloc(v, size * sizeof(double));

    // If reallocation failed, print error
    if (v == NULL) {
        printf("ERROR: Out of memory, realloced failed in grow\n");
        exit(0);
    }

    printf("New capacity: %d elements\n", size);
}
```

```
/**
* Shrink the dynamic array
*/
void Shrink()
{
    printf("Vector shrunk\n");
    printf("Previous capacity: %d elements\n", size);

    // Half the size of the vector and make it take up less space
    size /= 2;

    // Reallocate previous values and make the vector twice as small
    v = realloc(v, size * sizeof(double));

    // If reallocation failed, print error
    if (v == NULL) {
        printf("ERROR: Out of memory, realloced failed in shrink\n");
        exit(0);
    }

    printf("New capacity: %d elements\n", size);
}
```

```
/**
* Print the vector
*/
void PrintVector()
{

    // If there are no elements, print that vector is empty
    if (count == 0)
        printf("The dynamic array is empty!\n");

    // Print all of the doubles in the vector
    int j;
    for(j = 0; j < count; j++) {
        printf("%lf\n", v[j]);
    }
}

/**
* Add element to the vector
*/
void AddElement()
{

    // If more space is needed, grow the vector
    if(count == size)
        Grow();

    // Enter the new element
    double n;
    printf("Enter the new element: ");

    // Scan a double from the user
    scanf("%lf", &n);
    v[count] = n;
    count++;
}
```

```
/**
* Remove element to the vector
*/
void RemoveElement()
{

    // Cannot remove from an empty vector, check if so
    if(count == 0) {
        printf("ERROR: Cannot remove element from an empty dynamic array\n");
    }
    else {
        // Remove the last double
        v[count-1] = 0;

        // Lower the count
        count--;

        // Shrink if below 30%
        if(count < size * 0.3)
            Shrink();
    }
}
```

```
/**
* Insert an element into the vector
*/
void InsertElement()
{
    printf("Enter index of new value: ");

    int index;
    // Scan an index as a digit from the user
    scanf("%d", &index);

    // If index is out of bounds, print an error
    if(index > count || index < 0) {
        printf("ERROR: Index out of bounds\n");
    } else {
        // If the dynamic array is too small, grow it
        if(count > size)
            Grow();

        printf("Enter the new element: ");
        double value;

        // Scan a new element, as a double, from the user
        scanf("%lf", &value);

        /* Move the already existing values forward
        and insert the new value */
        int j;
        for(j = count; j > index; j--) {
            v[j] = v[j - 1];
        }
        v[index] = value;

        /* Increase the count since we added
        a new value */
        count++;
    }
}
```

```c
/**
* Print the menu and get a selection from the user
*
* @return Number representing a selection from a user
*/
int PrintMenu()
{
    int sel;

    printf("Main menu:\n\n" );
    printf("1. Print the array\n" );
    printf("2. Append element at the end\n" );
    printf("3. Remove last element\n" );
    printf("4. Insert one element\n" );
    printf("5. Exit\n\n" );
    printf("Select an option: " );

    // Scan a digit from the user
    scanf("%d", &sel);

    // Return the chosen digit
    return sel;
}
```

```
/**
* Run the main loop
*
* @return Number representing a selection from a user
*/
int Run()
{
    int sel;

    // While true
    while(true) {

        // Print the menu and get a selection
        sel = PrintMenu();

        // Next step depends on the selection made
        switch(sel) {

            // User chose 1
            case 1:
                printf("You selected \"Print the Array\"\n");
                PrintVector();
                break;

            // User chose 2
            case 2:
                printf("You selected \"Append element at the end\"\n");
                AddElement();
                break;

            // User chose 3
            case 3:
                printf("You selected \"Remove last element\"\n");
                RemoveElement();
                break;

            // User chose 4
            case 4:
                printf("You selected \"Insert one element\"\n");
                InsertElement();
                break;
```

```
            // User chose 5
            case 5:
                printf("You selected \"Exit\"\n");

                /* Return here, with no erros, to exit the function.
                Clean up will be next */
                return 0;

            // User chose soomething not on the menu
            default:
                printf("Please enter a valid number from the menu!\n\n");
                break;
        }

        printf("--------------------\n");
    }
}
```

```
/**
* Main function, argc and argv are unused in this case
*
* @param argc Number of arguments
* @param argv Array of argument character arrays
* @return Number representing a selection from a user
*/
int main (int argc, char *argv[])
{

    // Initalize the globals
    Initalize();

    // Run the loop
    Run();

    // Clean up
    Finalize();

    // Return with no errors
    return 0;
}
```

Running the Code:

```
Linok-2 :: Desktop/Embedded Des Enabling Robotics/lab1 » gcc lab1.c -o lab1
Linok-2 :: Desktop/Embedded Des Enabling Robotics/lab1 » ./lab1
Main menu:

1. Print the array
2. Append element at the end
3. Remove last element
4. Insert one element
5. Exit

Select an option:
```

```
Main menu:

1. Print the array
2. Append element at the end
3. Remove last element
4. Insert one element
5. Exit

Select an option: 1
You selected "Print the Array"
The dynamic array is empty!
--------------------
Main menu:

1. Print the array
2. Append element at the end
3. Remove last element
4. Insert one element
5. Exit

Select an option:
```

```
Select an option: 3
You selected "Remove last element"
ERROR: Cannot remove element from an empty dynamic array
--------------------
```

```
Select an option: 2
You selected "Append element at the end"
Enter the new element: 1
--------------------
```

```
Select an option: 2
You selected "Append element at the end"
Enter the new element: 2
--------------------
```

```
Select an option: 2
You selected "Append element at the end"
Vector grown
Previous capacity: 2 elements
New capacity: 4 elements
Enter the new element: 3
--------------------
```

```
Select an option: 1
You selected "Print the Array"
1.000000
2.000000
3.000000
--------------------
```

```
Select an option: 3
You selected "Remove last element"
--------------------
```

```
Select an option: 3
You selected "Remove last element"
Vector shrunk
Previous capacity: 4 elements
New capacity: 2 elements
--------------------
```

```
Main menu:

1. Print the array
2. Append element at the end
3. Remove last element
4. Insert one element
5. Exit

Select an option: 5
You selected "Exit"
Linok-2 :: Desktop/Embedded Des Enabling Robotics/lab1 »
```

Insert:

```
--------------------
Main menu:

1. Print the array
2. Append element at the end
3. Remove last element
4. Insert one element
5. Exit

Select an option: 1
You selected "Print the Array"
1.000000
2.000000
3.000000
4.000000
5.000000
--------------------
```

```
Select an option: 4
You selected "Insert one element"
Enter index of new value: 0
Enter the new element: 1123
--------------------
```

```
Select an option: 1
You selected "Print the Array"
1123.000000
1.000000
2.000000
3.000000
4.000000
5.000000
--------------------
```

```
Select an option: 4
You selected "Insert one element"
Enter index of new value: 6
Enter the new element: 67
--------------------
```

```
Select an option: 1
You selected "Print the Array"
1123.000000
1.000000
2.000000
3.000000
4.000000
5.000000
67.000000
--------------------
```

```
Select an option: 4
You selected "Insert one element"
Enter index of new value: 11
ERROR: Index out of bounds
--------------------
```