Storage Classes

30-11-25

Aim – Implement a program to demonstrate storage classes auto, extern, static, register dynamic memory allocation malloc, calloc, realloc.

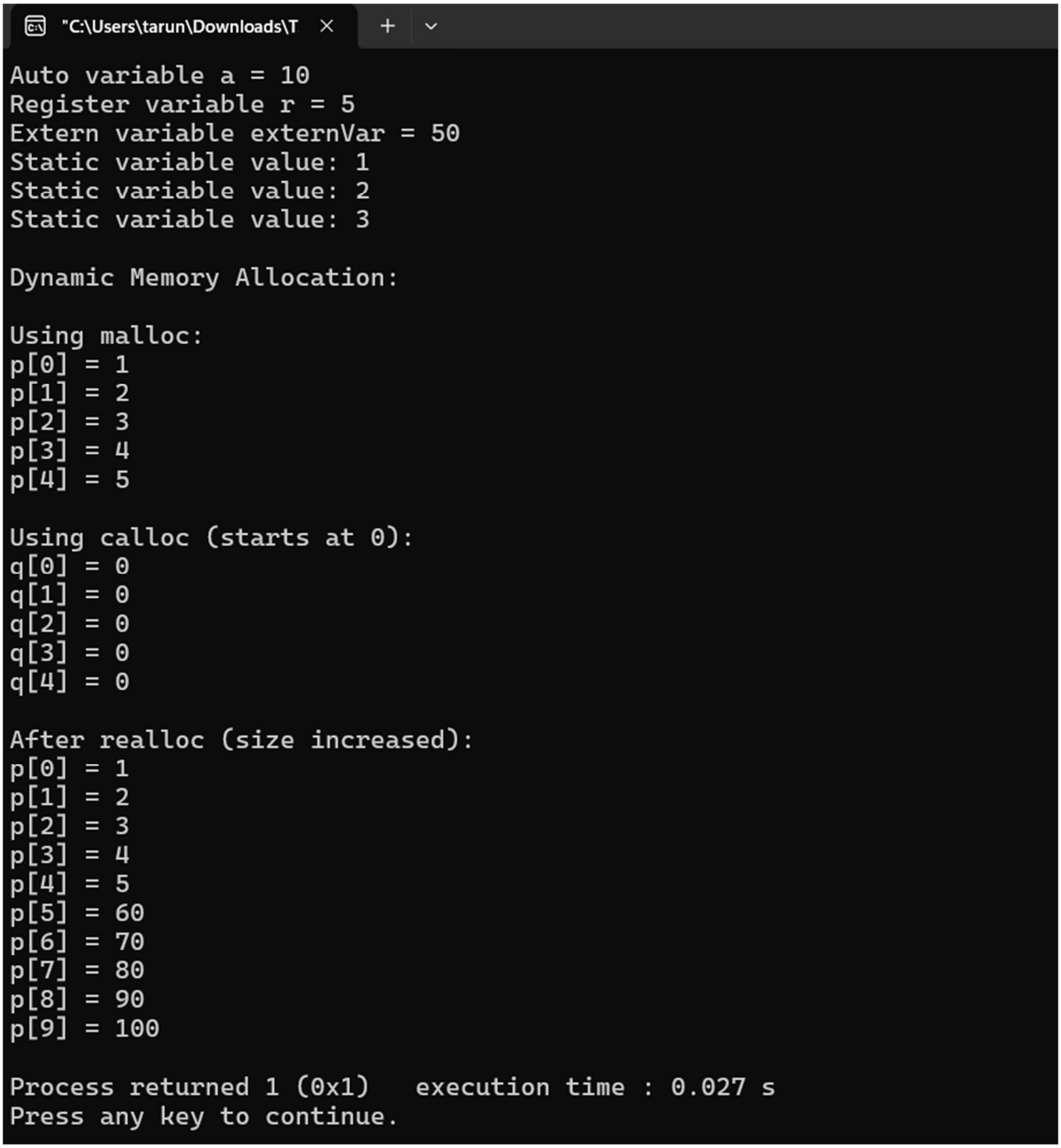
Theory –

In C, the auto storage class is the default for all local variables and such variables are stored on the stack, existing only during the execution of the function. The extern storage class is used when a variable is declared in one file and defined in another, allowing global variables to be shared across multiple source files without creating new memory. The static storage class allows a variable to retain its value between function calls, even though its scope may remain limited to the function. The register storage class suggests to the compiler that the variable should be stored in a CPU register for faster access, although the compiler may ignore this request. For dynamic memory management, malloc() allocates memory at runtime but does not initialize the allocated block. calloc() also allocates dynamic memory but initializes all allocated bytes to zero, making it suitable for arrays. realloc() is used to resize an already allocated memory block while preserving existing data when possible. Finally, dynamically allocated memory should always be released using free() to prevent memory leaks and ensure efficient memory usage.

A1.

```
#include <stdio.h>

#include <stdlib.h>


extern int externVar;

int externVar = 50;


void staticDemo() {

   static int count = 0;

   count++;

   printf("Static variable value: %d\n", count);

}


void main() {

   auto int a = 10;

   register int r = 5;


   printf("Auto variable a = %d\n", a);

   printf("Register variable r = %d\n", r);

   printf("Extern variable externVar = %d\n", externVar);
```

```
    staticDemo();
    staticDemo();
    staticDemo();

    int *p, *q;
    int n = 5;

    printf("\nDynamic Memory Allocation:\n");
    p = (int*)malloc(n * sizeof(int));
    printf("\nUsing malloc:\n");
    for (int i = 0; i < n; i++) {
        p[i] = i + 1;
        printf("p[%d] = %d\n", i, p[i]);
    }
    q = (int*)calloc(n, sizeof(int));
    printf("\nUsing calloc (starts at 0):\n");
    for (int i = 0; i < n; i++) {
        printf("q[%d] = %d\n", i, q[i]);
    }
    p = (int*)realloc(p, 10 * sizeof(int));
    printf("\nAfter realloc (size increased):\n");
    for (int i = 5; i < 10; i++) {
        p[i] = (i + 1) * 10;
    }
    for (int i = 0; i < 10; i++) {
        printf("p[%d] = %d\n", i, p[i]);
    }
    free(p);
    free(q);
}
```



Conclusion

I learned how C storage classes control the scope and lifetime of variables, and how each class behaves differently. I also understood how dynamic memory functions like malloc(), calloc(), and realloc() allocate and manage memory at runtime, and why using free() is important to avoid memory leaks.