



SIMON FRASER UNIVERSITY
ENGAGING THE WORLD

CMPT 125 - Introduction to Computing Science and Programming II - Fall 2022

Lab 5. Dynamic memory allocation and measuring performance

- Local Variables:
 - Variables which are declared inside of a scope.
- Global Variables:
 - Variables which are declared outside of any function.
- Static variables:
 - Variables which are declared inside of a scope but keeps its value.

Reference:

<https://overiq.com/c-programming-101/local-global-and-static-variables-in-c/>

- What about the following cases:
 - We don't know the size/number of the variables that we want
 - When we create a local variable and we want to have an access to it from outside of the scope.

- Reference:

- <https://en.cppreference.com/w/c/memory/malloc>

- Function:

```
void* malloc( size_t size );
```

- Input: number of bytes to allocate
- Return: the pointer to the beginning of newly allocated memory or a null pointer.

- Reference:

- <https://en.cppreference.com/w/c/memory/free>

- Function:

```
void free( void* ptr );
```

- Input: pointer to the memory to deallocate

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

int main(void)
{
    int *p1 = malloc(4*sizeof(int)); // allocates enough for an array of 4 int
    int *p2 = malloc(sizeof(int[4])); // same, naming the type directly
    int *p3 = malloc(4*sizeof *p3);   // same, without repeating the type name

    if(p1) {
        for(int n=0; n<4; ++n) // populate the array
            p1[n] = n*n;
        for(int n=0; n<4; ++n) // print it back out
            printf("p1[%d] == %d\n", n, p1[n]);
    }

    free(p1);
    free(p2);
    free(p3);
}
```

- Run “get_name.c” and “get_name_wrong.c” and check what is the error in the wrong version.
- Run “random_ar.c” which is for allocating as much memory as needed.
- You can see more samples and read about “calloc” in this link:
 - <https://www.programiz.com/c-programming/c-dynamic-memory-allocation>

- Function:

```
int gettimeofday ( struct timeval *tp , struct timezone *tz )
```

- The current time is expressed in elapsed seconds and microseconds since 00:00:00, January 1, 1970 (Unix Epoch).
- The 1st argument points to the timeval structure. The timeval structure is declared as below in sys/time.h header file :

```
struct  timeval {  
    time_t      tv_sec ; //used for seconds  
    suseconds_t tv_usec ; //used for microseconds  
}
```

- Return: On success, the gettimeofday() return 0, for failure the function returns -1.
- Reference:
 - https://linuxhint.com/gettimeofday_c_language/

- “measure_time.c” allows measuring time of execution.
 - Compare the running time of the two functions for different values of n .
 - Compute the actual running time by subtracting time before and time after.
 - Compare the running time of different implementations of Fibonacci functions we saw in lecture 8.