Laboratory 1: Review Exercises in C

CSC205A Data structures and Algorithms Laboratory B. Tech. 2015

Vaishali R Kulkarni

Department of Computer Science and Engineering
Faculty of Engineering and Technology
M. S. Ramaiah University of Applied Sciences
Email: vaishali.cs.et@msruas.ac.in
Tel: +91-80-4906-5555 (2212) WWW: www.msruas.ac.in

Introduction and Purpose of Experiment

- To get familiar with the data types and local variables and random number generation.
- Basic concepts such as data types and local variables are part and parcel of almost all the c programs. Hence sound knowledge is most essential in this regard.
- The random number generation essential for many applications, for ex. rolling a dice for many in gaming applications such as backgammon which requires a random number generation from 1 to 6.

Aim and objectives

Aim:

- To design and develop a C programs to understand Data types and its manipulation, local variables and its characteristics. Also to generate random numbers and to familiarize with its applications.
- To demonstrate the use and significate of the above concepts in C programming.

Objectives:

At the end of this lab, the student will be able to

- Use variables of the basic data types with proper declarations
- Validate input data
- Generate random numbers for any application

Theory Behind the Experiment

Program Parts

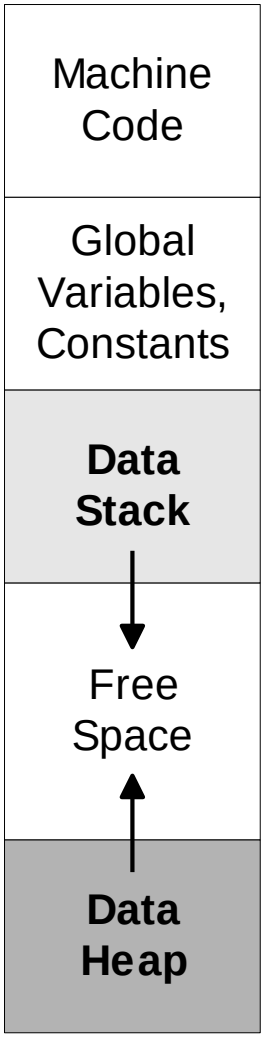
Space for program code includes space for machine language code and data

Data broken into:

- space for global variables and constants
- data stack - expands/shrinks while program runs
- data heap - expands/shrinks while program runs

Local variables in functions allocated when function starts:

- space put aside on the data stack
- when function ends, space is freed up
- must know size of data item (int, array, etc.) when allocated (*static allocation*)



Theory Behind the Experiment

- Stack memory:
  - It's a special region of your computer's memory that stores temporary variables created by each function (including the main() function).
  - Every time a function declares a new variable, it is "pushed" onto the stack. Then every time a function exits, all of the variables pushed onto the stack by that function, are freed
- Heap memory:
  - The heap is a region of your computer's memory that is not managed automatically for you
  - It is a more free-floating region of memory (and is larger).
  - To allocate memory on the heap, you must use malloc() or calloc(), which are built-in C functions.
  - Once you have allocated memory on the heap, you are responsible for using free() to deallocate that memory once you don't need it any more.

Theory Behind the Experiment(2)

Memory Management Functions:

- malloc: The most important storage allocation function is named malloc. When called, malloc allocates a block of size bytes and returns a pointer to it: void *malloc(size_t size);

  malloc returns a "generic" pointer that is capable of pointing at an object of any type. malloc's return value can be stored in a variable of any pointer type.

- calloc: An array can be dynamically allocated by a call of the calloc function. calloc is similar to malloc, but allocates space for an array with nmemb elements, each of which is size bytes long:

  void *calloc(size_t nmemb, size_t size);

- When a block of memory is no longer needed, it can be released by calling the free function:

```
int *p,*q;
p = malloc(sizeof(int));
free(p);
```

Theory Behind the Experiment(3)

Random Number Generation

- C function rand() generate a random number and it is a function of stdlib.
- The function rand() returns a pseudo-random integral number.
  This number will be in the range 0 to RAND_MAX.
- The algorithm of rand() uses a seed to generate the series of numbers, this is why srand must be used to initialize the seed to some distinctive value.

Experimental Procedure

- Analyse the problem statement
- Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
- Implement the algorithm in C language
- Compile the C program
- Design test cases and test the implemented program
- Document the Results
- Analyse and discuss the outcomes of your experiment

Questions

Demonstrate the use of data types, local variables and Random numbers by designing appropriate algorithms for the below problems. Tabulate the output for various inputs and verify against expected values. Analyse the efficiency of the algorithm. Describe your learning along with the limitations of overall approach if any. Suggest how these can be overcome.

- Write a C program to illustrate random number generation.
- Write a C program to find sum of n elements, allocate memory dynamically using malloc() and calloc() function.
- Write a program to reverse an array iteratively and recursively separately and compare the results.
- Write a program to demonstrate the stack and heap memory allocation.

Key factors and discussion

- Illustration of Random Number Generation
  - Random numbers of different range should be generated. How to modify from one range to another range have to be discussed. For ex. Generating random number between: (1) 1 to 6, (2) 150 to 250 ,(3) -25 to 25 and so on.
  - How to produce true random numbers without repeations
- Dynamic memory allocation using calloc and malloc
  - Allocate memory for different type of variables such as int, char and float.
  - Type casting between void pointer and others
  - Difference betwen calloc and malloc in terms of usage, outcome and significance have to be considered
- Recursion and iteration
  - Finding the average of array elements using both recursion and iteration.
  - Compare the efficiency of both the techniques. Compare factors like memory usage and limitation in applications, ease of usage and so on.
  - Realize the above facts thru a c program.

Expected Outputs

Random Number generation

```
Guess the number (1 to 10): 2
The number is higher
Guess the number (1 to 10): 3
The number is higher
Guess the number (1 to 10): 5
The number is higher
Guess the number (1 to 10): 4
The number is higher
Guess the number (1 to 10): 6
The number is higher
Guess the number (1 to 10): 9
The number is lower
Guess the number (1 to 10): 8
That is correct!
```

Expected Outputs

Reversing an array using recursion

```
Given array is  : 1      2      3      4      5
reverse array is: 5      4      3      2      1
```

Reversing an array using recursion

```
Given array is  : 1      2      3      4      5
reverse array is: 5      4      3      2      1
```

Results and Presentations

- Calculations/Computations/Algorithms

  The calculations/computations/algorithms involved in each program has to be presented

- Presentation of Results

  The results for all the valid and invalid cases have to be presented

- Analysis and Discussions

  how the data is manipulated or transformed, what are the key operations involved. Errors encounters and how they are resolved.

- Conclusions

  Summary

Comments

- Limitations of Experiments

  Outline the loopholes in the program, data structures or solution approach.

- Limitations of Results

  Present the test cases; justify if the program is tested correctly considering all the outcomes. Mention what is not tested, if any.

- Learning happened

  What is the overall learning happened

- Conclusions

  Summary

References

- Gilberg, R. F., and Forouzan, B. A. (2007): A Pseudocode Approach With C, 2nd edn. Cengage Learning
- The algorithm for recursive level order traversal is taken from:

  http://www.geeksforgeeks.org/level-order-tree-traversal/