(Lab 6) C Programming - Arrays, Pointers and Functions

CS2013 Systems Programming

Department of CSE, IIT Palakkad

IIT Palakkad

Sep 04, 2025

Quiz 6 (15 minutes, **Do not copy the question**)

In `Raj` architecture computer, every integer is 6 bits (Recall, this was 32 bits for non-`Raj` architectures).

1. What is the largest value of a signed integer possible in this architecture ?
2. Consider the following C code written in `Raj` architecture.

   ```
   int a = 18, b = 14;
   int c = a + b;
   ```

   What will be the value stored in the variable `c` ?
3. Suppose, you do not know the values of `a` and `b`. Consider the integer value corresponding to $(a + b)/2$.

   This quantity will be smaller than the value in question 1. Write a C code to compute it *correctly* irrespective of the value of `a` and `b` in a `Raj` machine.

Plan (Demo)

- Arrays - declaration, usages and examples
- From arrays to pointers
- Arrays of various types
- Strings and operations
- Array of pointers as two dimensional arrays
- Functions in C language
- Function calling mechanism (Call by value and Call by reference)

Why use & in scanf ?

- What happens in a call to `scanf()` ?

Why use & in scanf ?

- What happens in a call to `scanf()` ?
- OS uses I/O and gets the input (Why do we need OS ?)

Why use & in scanf ?

- What happens in a call to `scanf()` ?
- OS uses I/O and gets the input (Why do we need OS ?)
- OS places contents read in the variable

Why use & in scanf ?

- What happens in a call to `scanf()` ?
- OS uses I/O and gets the input (Why do we need OS ?)
- OS places contents read in the variable
- To do this, address of the memory location is needed.
- `x` is a variable
  - `&x` returns address of `x` in memory
  - Address returned is a *logical* address.
  - OS translates it to *physical* address before putting the content.

Why use & in scanf ?

- What happens in a call to `scanf()` ?
- OS uses I/O and gets the input (Why do we need OS ?)
- OS places contents read in the variable
- To do this, address of the memory location is needed.
- `x` is a variable
  - `&x` returns address of `x` in memory
  - Address returned is a *logical* address.
  - OS translates it to *physical* address before putting the content.
- What happens in `printf()` ? Why is this **not** needed ?

Pointers

- Pointer is address of a data stored in memory location.
- For each type of data, we have a separate pointer
- Examples
  - `int * x` - for storing address of another variable which stores an integer.
  - `float * y` - for storing address of another variable which stores a float.
- How to get the address of a variable ?
  - `&` – also called as address operator

Pointers

- Pointer is address of a data stored in memory location.
- For each type of data, we have a separate pointer
- Examples
  - `int * x` - for storing address of another variable which stores an integer.
  - `float * y` - for storing address of another variable which stores a float.
- How to get the address of a variable ?
  - `&` – also called as address operator
- How to read the contents that a pointer is pointing to ?
  - `*` – also called as indirection operator

Quick Summary

- Arrays - how to use and abuse
- Pointers - why use it ? how to use it ?
- Strings - array of characters
- Functions - call by value and call by reference
- `size_t` type. Need for type casting

Lab Exercise

Questions (Do the following in your repo)

- Do `$ git switch lab05`. Commit all the changes.
- Do `$ git push -u origin lab05`

Lab Exercise

Questions (Do the following in your repo)

- Do `$ git switch lab05`. Commit all the changes.
- Do `$ git push -u origin lab05`
- Do `$ git fetch && git merge`
- Do `$ git switch lab06` to see the `questions.md` in `lab06` folder.
- **To push changes: do `$ git push -u origin lab06`**

Class repo (for in-class demo)

- Accessible via
  - `git clone git@gitserver:class_repo`
- To see latest changes, cd to the class_repo and do
  - `git fetch && git merge`
  - This does a git pull

Class has ended

- No more pushes to `gitserver`.
- Complete the exercises during off-lab hours.

Humble Request

Please keep the chairs in position before you leave.
(as a token of respect for our CFET staff)