

# Short-term Hands-on Supplementary Course on C Programming



## **SESSION 9: Pointers**

**NIVEDHITHA D**  
**KARTHIK D**

Time: 6:30 - 8:00 PM  
Date: June 22th, 2022  
Location: Online



# Agenda

1. Administrative Instructions
2. What are Pointers?
3. Declaring and using Pointers
4. Pointer Arithmetic
5. Double Pointers
6. Pointers and Arrays
7. Static vs. Dynamic Memory Allocation
8. Dynamic Memory Allocation in C
  - a. Primitive Types
  - b. Arrays and Strings
  - c. Functions
9. Tutorial: Arrays and Pointers
10. Next Session

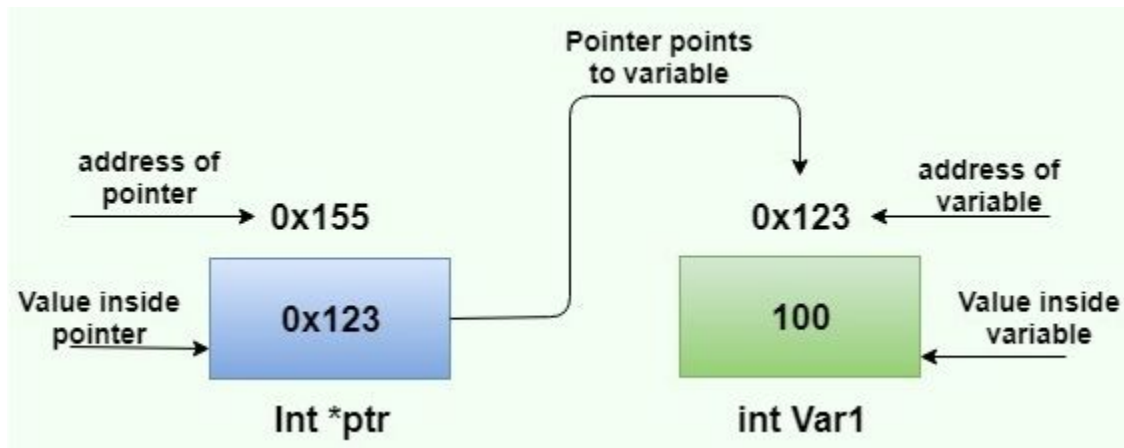
# Administrative Instructions

- Please fill out the feedback form - will be shared in the chat
- Join us on Microsoft Teams,  
Team Code: **rzlaicv**

**GITHUB REPOSITORY!** 

# What are Pointers?

A *pointer variable* (or *pointer* in short) is basically the same as the other variables, which can store a piece of data. Unlike normal variable which stores a value (such as an `int`, a `double`, a `char`), a *pointer stores a memory address*.



# Declaring and using Pointers

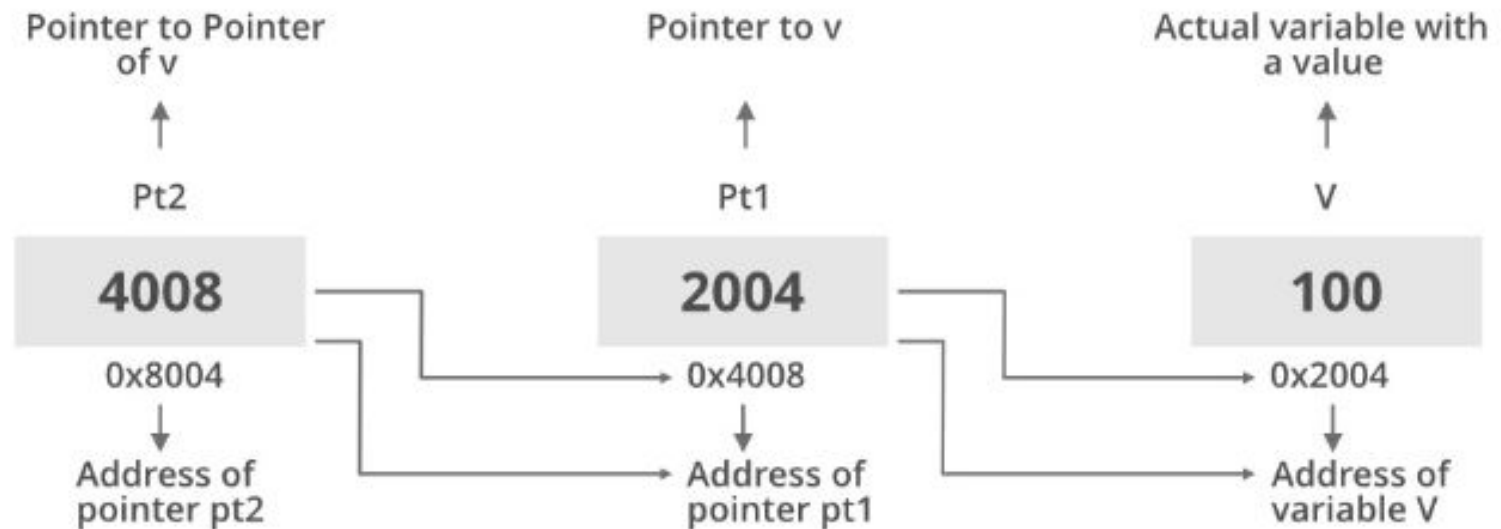
Computer		Programmers		
Address	Content	Name	Type	Value
90000000	00	sum	int (4 bytes)	000000FF (255 <sub>10</sub> )
90000001	00			
90000002	00			
90000003	FF			
90000004	FF	age	short (2 bytes)	FFFF (-1 <sub>10</sub> )
90000005	FF			
90000006	1F			
90000007	FF	average	double (8 bytes)	1FFFFFFFFFFFFFFF (4.45015E-308 <sub>10</sub> )
90000008	FF			
90000009	FF			
9000000A	FF			
9000000B	FF			
9000000C	FF			
9000000D	FF			
9000000E	90			
9000000F	00	ptrSum	int* (4 bytes)	90000000
90000010	00			
90000011	00			
90000012	00			

Note: All numbers in hexadecimal

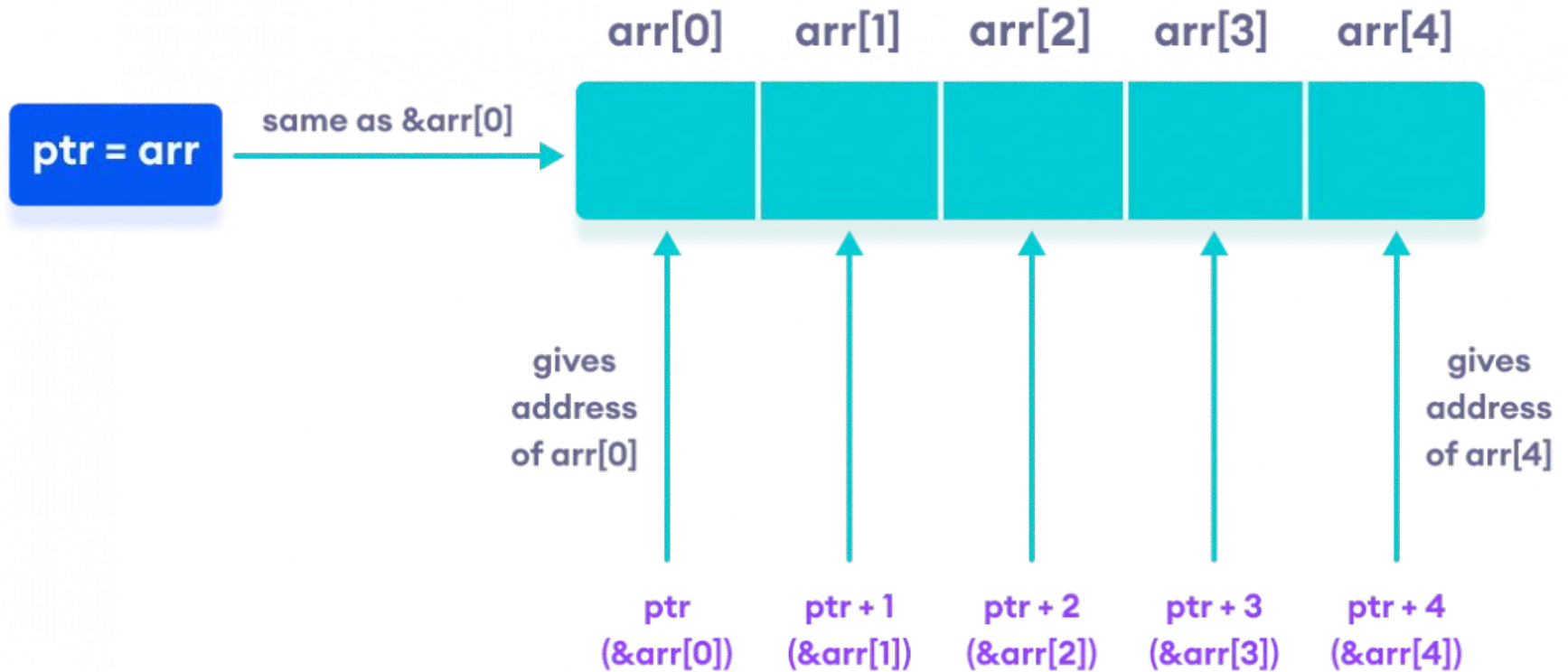
```
type *ptr;
// or
type* ptr;
// or
type * ptr;
```

```
1  #include <stdio.h>
2
3  int main(void) {
4      int sum = 255;
5      short age = -1;
6      double average =
7          4.45015E-308;
8      int* ptrSum = &sum;
9  }
```

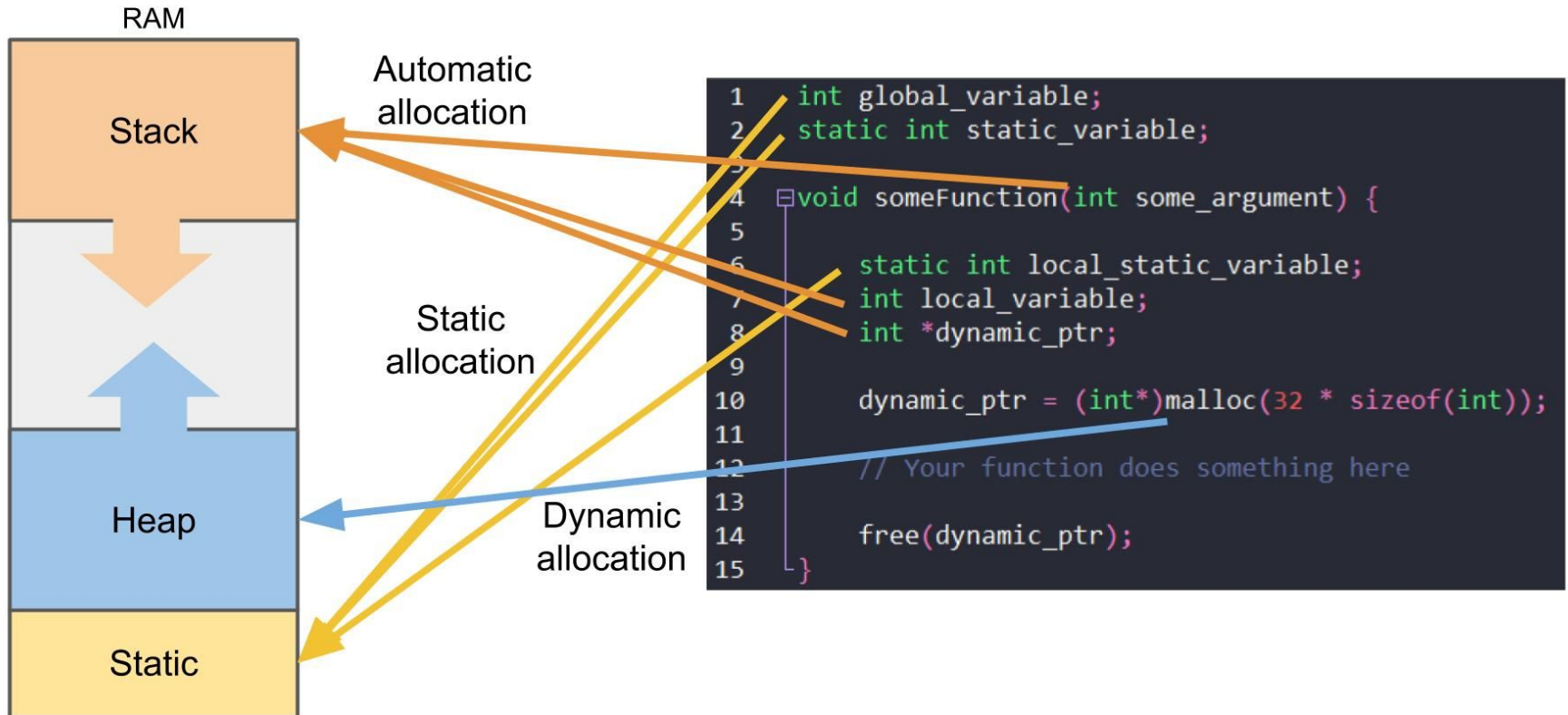
# Double Pointers



# Pointers & Arrays



# Memory Allocation in C

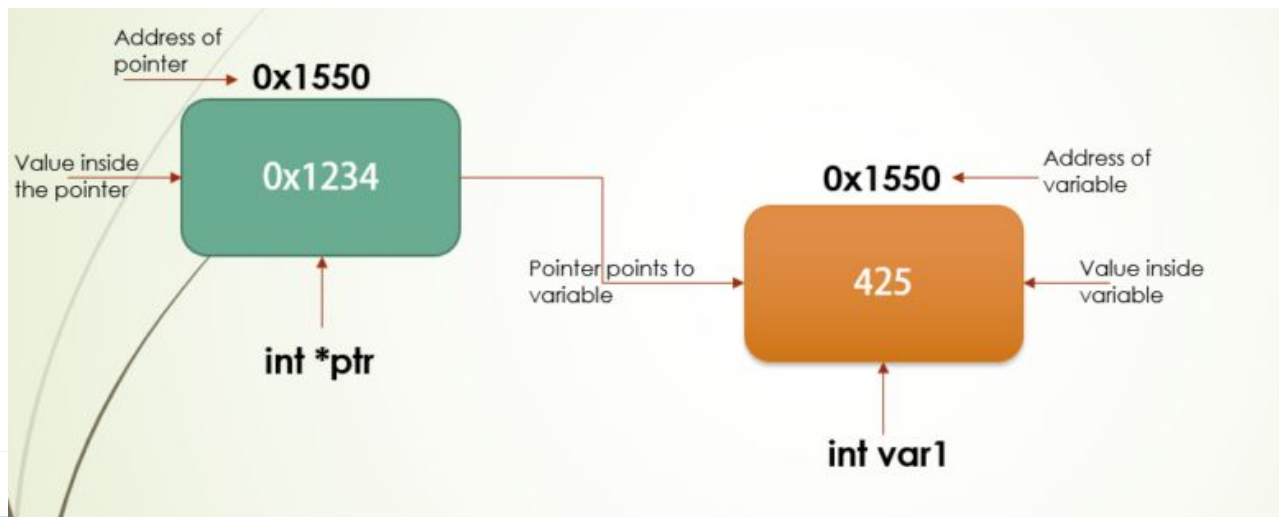




# Static vs. Dynamic Memory Allocation

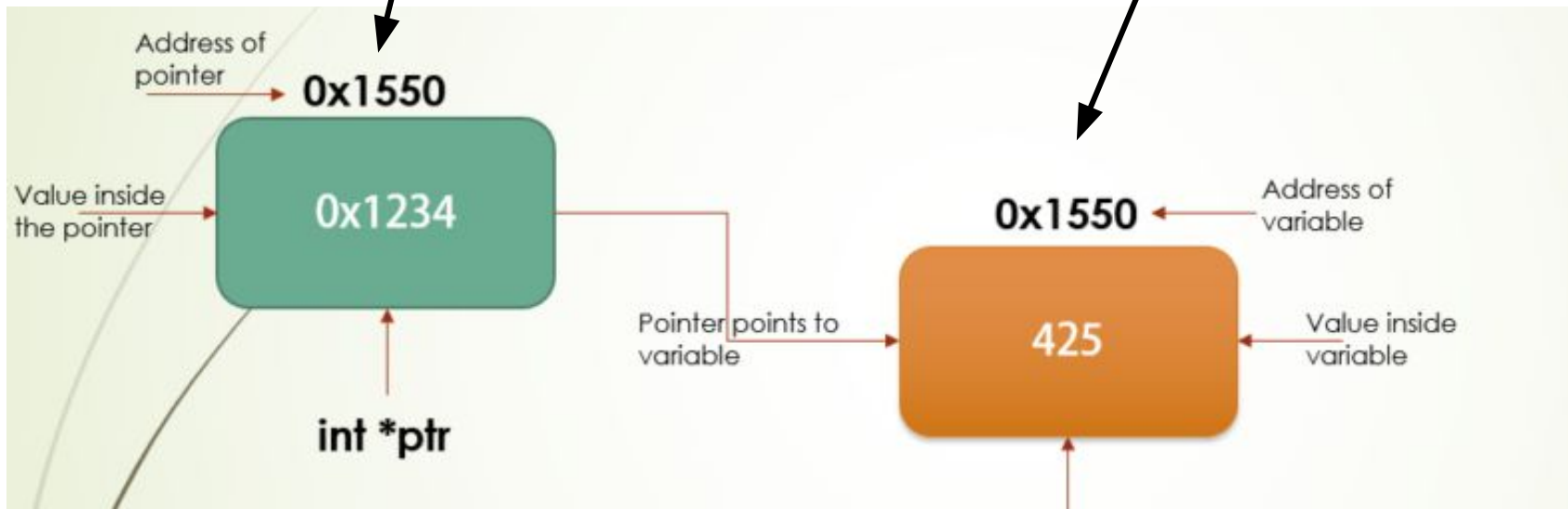
Dynamic Memory	Static Memory
Allocated at run time	Allocated at compile time
Memory can be altered during program execution	Memory cannot be altered during program execution
Example: Linked list	Example: Array

- The **heap** is often called **unnamed variable space**

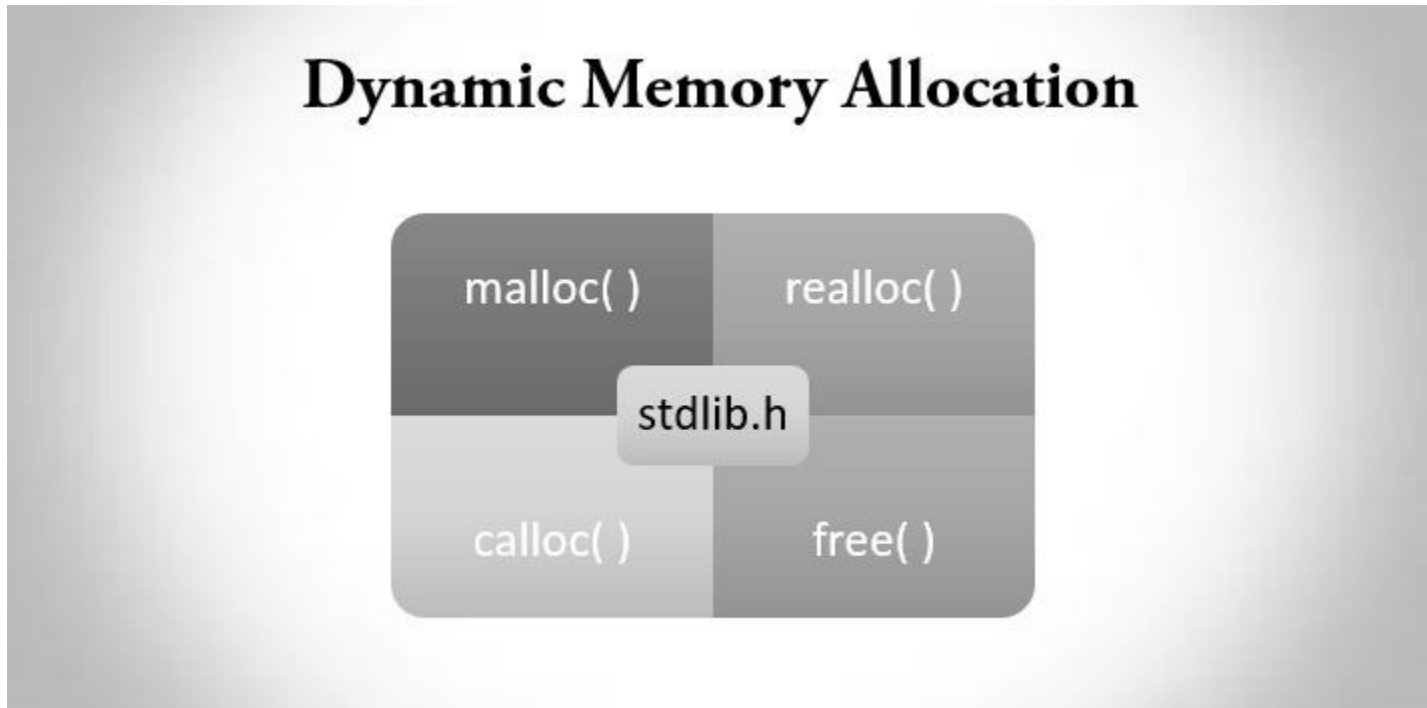


This is in  
**Stack/Global**

This is in **Heap**  
(if dynamically allocated)



# Dynamic Memory Allocation in C



## Syntax:

- `void *malloc( size_t size );`
- `void *calloc( size_t num, size_t size );`
- `void *realloc( void *ptr, size_t new_size );`
- `void free( void* ptr );`

# TUTORIAL

## Creating and Returning Pointers from Functions

```
int create_array(int size){
    int *arr = malloc(5*sizeof(int));

    int arr1[] = {1, 2, 3, 4, 5}; /// int *arr -> start addr of array

    for(int i=0;i < 5; i++){
        *(arr+i) = *(arr1+i);
    }

    return *(arr);
}
```

# Next Session

**MORE POINTERS!!!**

C isn't hard:

```
void (*(*f[]))()()
```

defines *f* as an array of unspecified size of pointers to functions that return pointers to functions that return void.

Any Questions