# Stacks

A subset of a link list

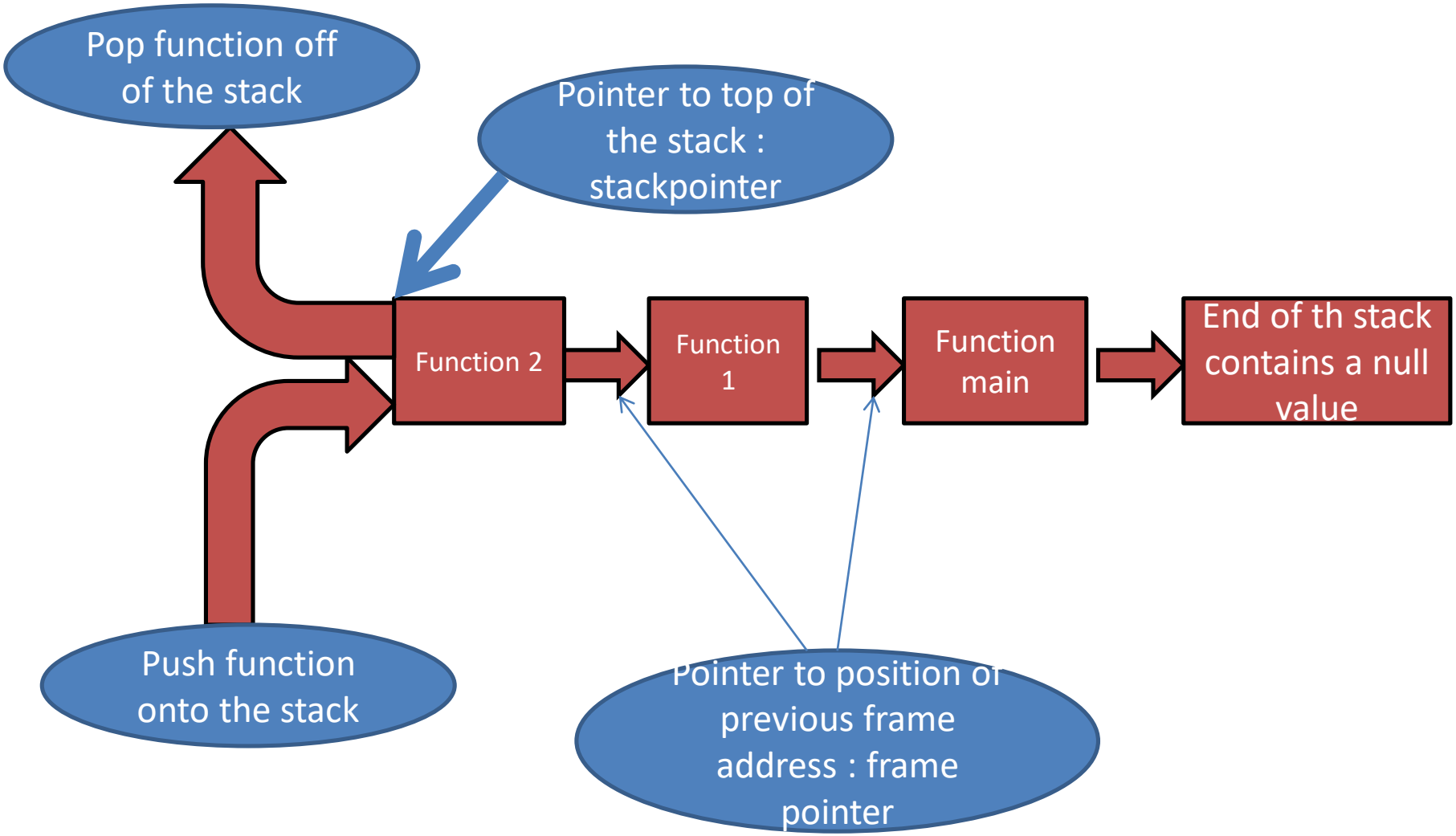
# Introduction

- A stack and how it is implemented
  - A LIFO
  - The push (add node  to stack)
  - pop functions (delete note from stack)

- A queue and how  it is implemented
  - A FIFO
  - The enqueue function (add node to queue)
  - Dequeue function (delete node from the queue)

# The Stack Data Structures

- A stack implemented using pointers?



# 12.5.3 Applications of Stacks

- Stacks have many interesting applications.
- For example, whenever a *function call* is made, the called function must know how to *return* to its caller, so the *return address* is pushed onto a stack.
- If a series of function calls occurs, the successive return values are pushed onto the stack in *last-in, first-out order* so that each function can return to its caller.

# A call Stack

- A call stack is a data structure that is used to is used to store information about the active routines (functions) in a program.

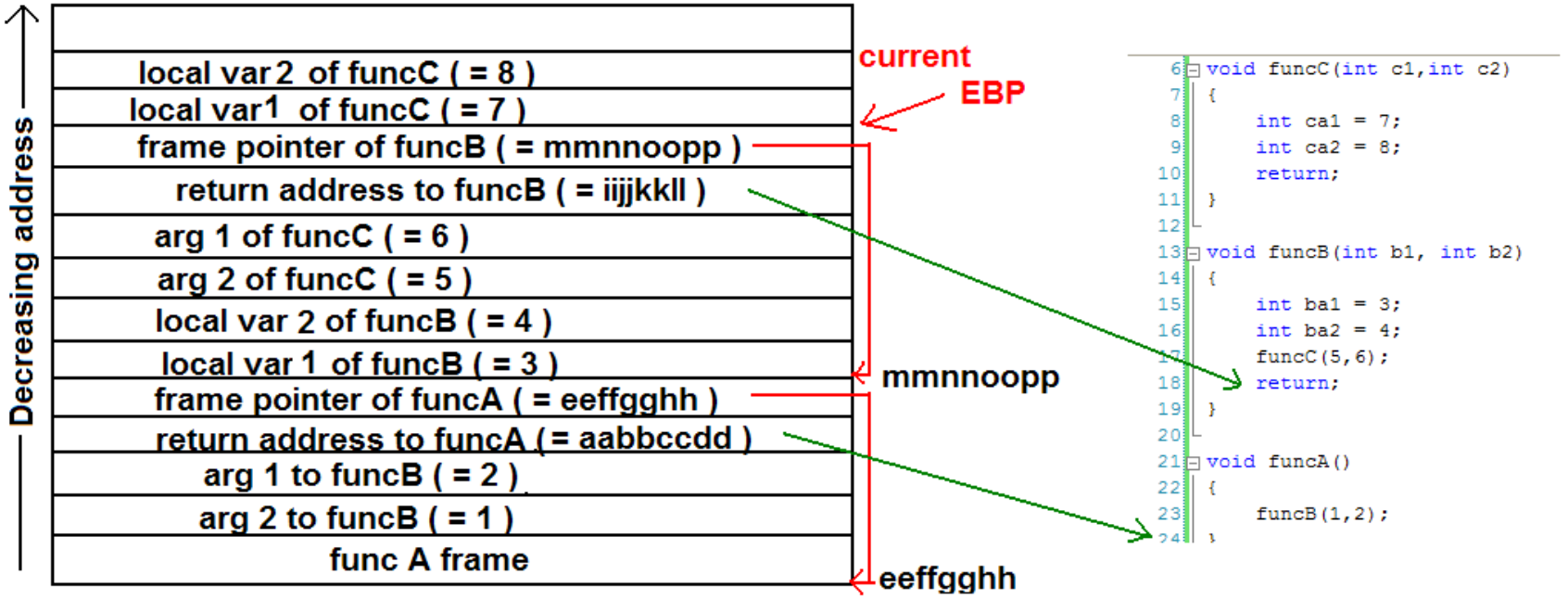
- Each time a function (main or other functions are invoked they, including local variables…) are **pushed** onto the stack.
- Once the functions is finished executing “it” is **popped** from the stack.

- A stack operates on *the last in first out* principle (LIFO) (*push* to beginning of the stack and *pop* from the beginning of the stack.

- This procedure is controlled by the use of a stack pointer which keeps track of the “starting” position, sometimes referred to as the head/top, of the stack.

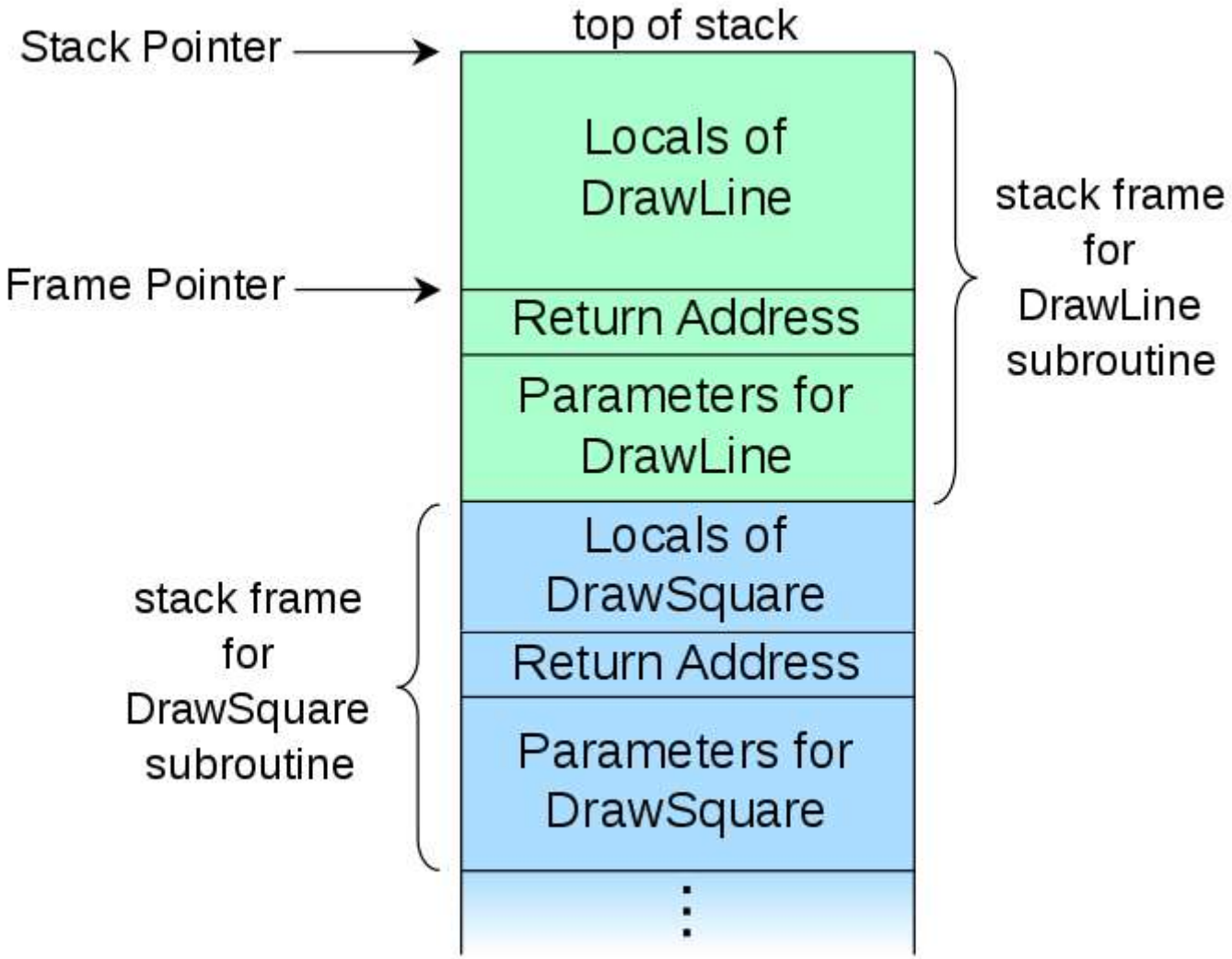
# The Call stack

- Example of call stack:




EBP: extended base pointer: aka frame pointer
ESP extended stack pointer: stack pointer

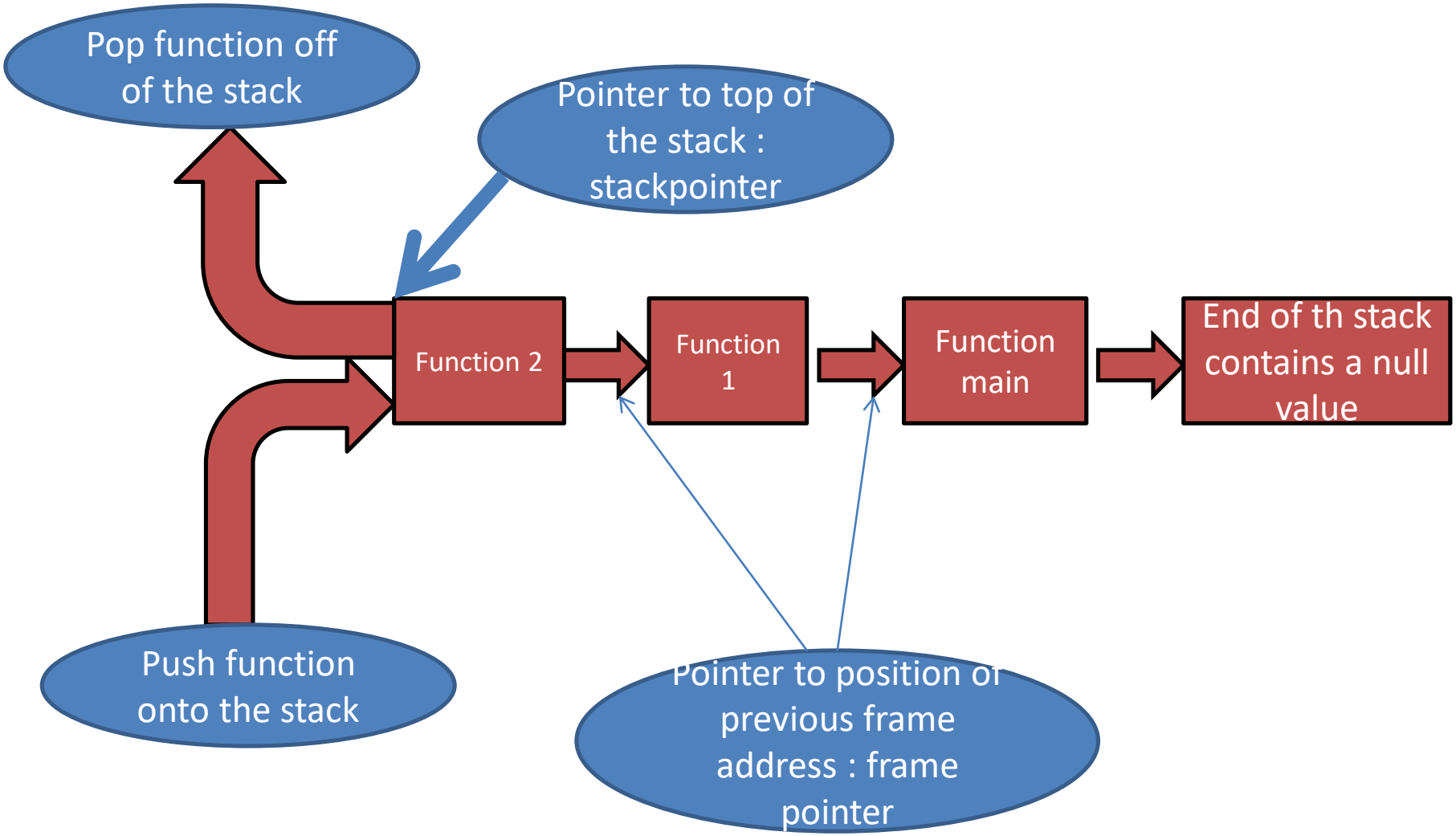
# Call stack: Stack and Frame pointer



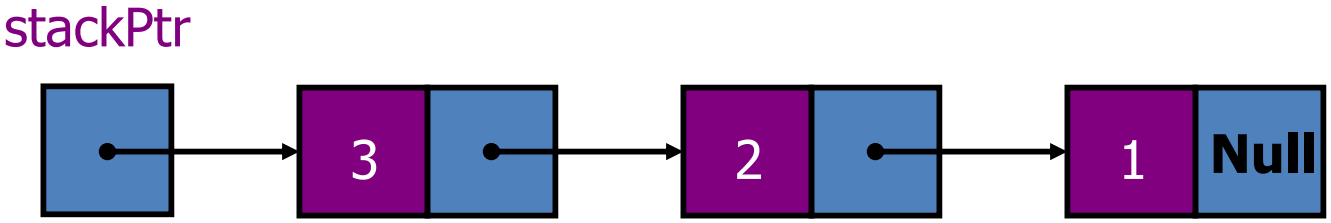

call stack Wikipedia

# The Stack Data Structures

- A stack implemented using pointers?



# Implementation of a Stack:




- A *stack* is a series of connected *nodes*
- Each node contains at least
  - A piece of data (e.g. P.C.B or functions)
  - Pointer to the next node in the stack ;

- *stackPtr*: pointer to the first node
- The last node points to `NULL`
- `stackPtr = Null indicates an empty stack`

# Create Node

- Define node structure
  - 
  - struct stackNode {
  - **int data; // define data as an int**
  - **struct stackNode *nextPtr; // stackNode pointer**
  - };
  - *typedef struct stackNode StackNode*

- Declare and create a node:
  - StackNode newPtr = malloc(sizeof(StackNode));

- Assign value to **StackNode** fields
  - newPtr -> data = 4;
  - newPtr -> nextPtr = NULL;

# Implementation of a stack

- Two basic functions associated with stacks:
  - The **push** function: this adds a “node” to the beginning of a stack
  - The **pop** function: removes a “node” from the beginning of a non-empty stack.

- Implementation is a combination of:
  - Creating a new node; (**in C use malloc**)
  - Assigning data to the data element of the node
  - Using the **push function** add a node to the stack

  - In the pop function you pop (delete) a node
  - The *constrain* that you *can not pop* from an *empty stack*.

# Main function Stack.c

denis.manley@apollo: ~/OS2/Week5_stack_queue

```
int main(void)
{
   StackNode* stackPtr = NULL; // points to stack top
   int value; // int input by user

   instructions(); // display the menu
   printf("%s", "? ");
   unsigned int choice; // user's menu choice
   scanf("%u", &choice);

   // while user does not enter 3
   while (choice != 3) {

      switch (choice) {
         // push value onto stack
         case 1:
            printf("%s", "Enter an integer: ");
            scanf("%d", &value);
            push(&stackPtr, value);
            printStack(stackPtr);
            break;
         // pop value off stack
         case 2:
            // if stack is not empty
//             if (!isEmpty(stackPtr))
            if (stackPtr != NULL)
            {
               printf("The popped value is %d.\n", pop(&stackPtr));
            }

            printStack(stackPtr);
            break;
         default:
            puts("Invalid choice.\n");
            instructions();
            break;
      } // end switch

      printf("%s", "? ");
      scanf("%u", &choice);
                                                    34,1          25%
```

# The Function push

- The function consists of three steps:
  1. Create a new node using malloc and assign the location of the allocated memory to `newPtr`
  2. Assign `data, that is to be added to the stack,` to the new node; Assign the pointer part of the new node to store the address in stackPtr; the address of the node that is currently top of the stack.
  3. Move stack`Ptr` to point to the new node. {assign address of new node to the stackPtr}

# Sample code push function

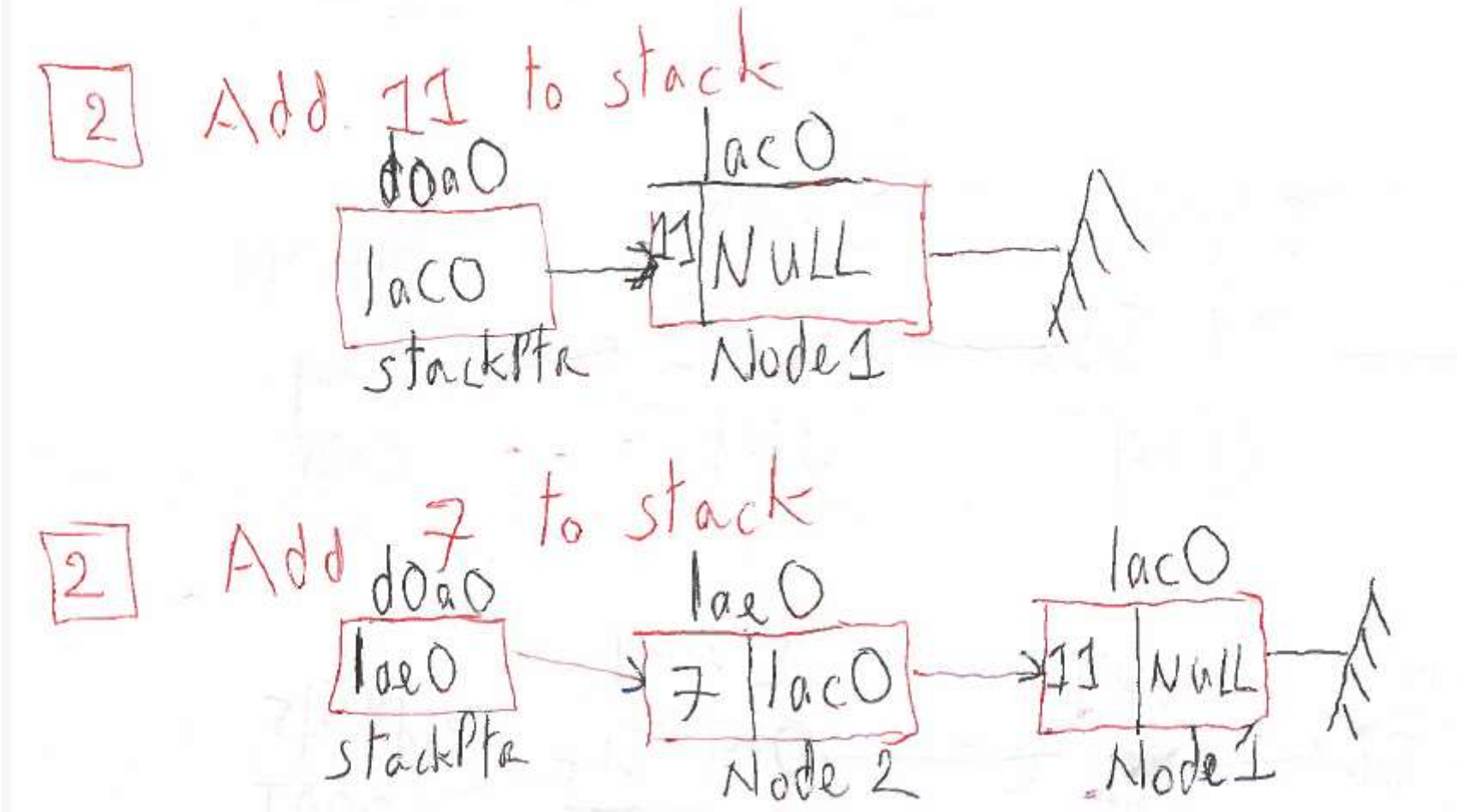
```
// insert a node at the stack top
void push(StackNode* *topPtr, int info)
{
   StackNode* newPtr = malloc(sizeof(StackNode));

   // insert the node at stack top
   if (newPtr != NULL) {
      newPtr->data = info;
      newPtr->nextPtr = *topPtr;
      *topPtr = newPtr;
   }
   else { // no space available
      printf("%d not inserted. No memory available.\n", info);
   }
}
```

# Add 3 nodes to empty stack: stack2.c

```
denis.manley@soc-apollo:~/OS2/Week5_stack_queue$ ./Stack2
the address of the stackPtr is 0x7ffe3bd1d0a0
Enter choice:
1 to push a value on the stack
2 to pop a value off the stack
3 to end program
? 1
Enter an integer: 11
****************** value of stack and frame pointers after the push ***********************

the value of the stack pointer (stackPtr)  is: 0x55adcbc81ac0
the value of the nextPtr in top or first node is: (nil)
********************************************************************************

The stack is:
11|(nil) (0x55adcbc81ac0) --> NULL

Enter choice:
1 to push a value on the stack
2 to pop a value off the stack
3 to end program
? 1
Enter an integer: 7
****************** value of stack and frame pointers after the push ***********************

the value of the stack pointer (stackPtr)  is: 0x55adcbc81ae0
the value of the nextPtr in top or first node is: 0x55adcbc81ac0
********************************************************************************

The stack is:
7|0x55adcbc81ac0 (0x55adcbc81ae0) --> 11|(nil) (0x55adcbc81ac0) --> NULL

Enter choice:
1 to push a value on the stack
2 to pop a value off the stack
3 to end program
? 1
Enter an integer: 12
****************** value of stack and frame pointers after the push ***********************

the value of the stack pointer (stackPtr)  is: 0x55adcbc81b00
the value of the nextPtr in top or first node is: 0x55adcbc81ae0
********************************************************************************

The stack is:
12|0x55adcbc81ae0 (0x55adcbc81b00) --> 7|0x55adcbc81ac0 (0x55adcbc81ae0) --> 11|(nil) (0x55adcbc81ac0) --> NULL

Enter choice:
1 to push a value on the stack
2 to pop a value off the stack
3 to end program
?
```

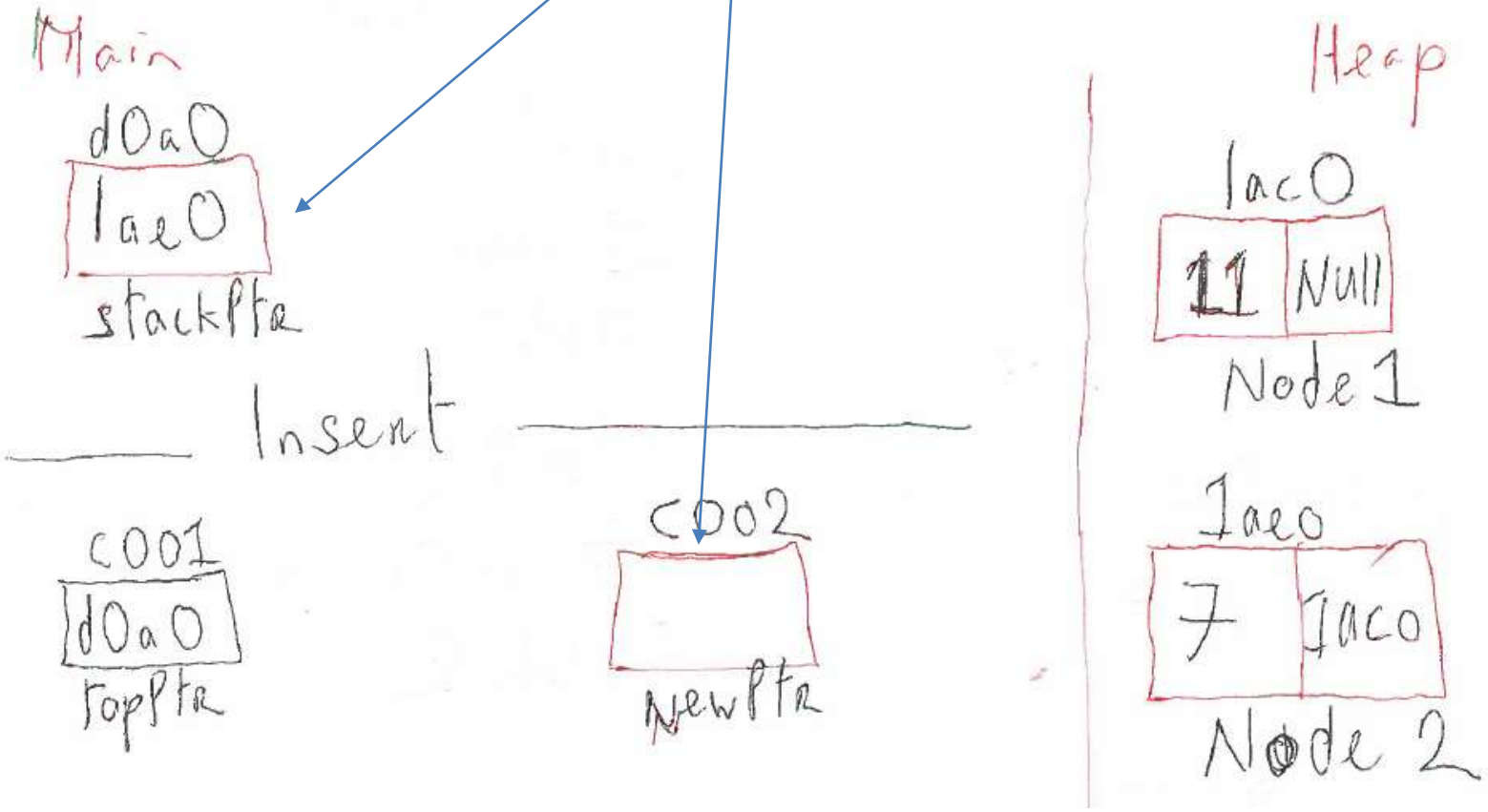
# Push on to stack

- Before and after: representation of stack



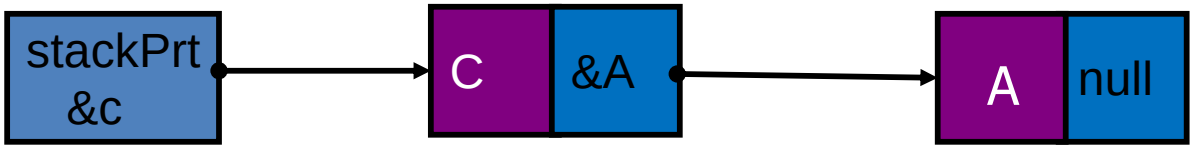
# Illustration of push on to stack

- Contents of memory locations of relevant variables: Before the push onto the stack. What happens when a new value 12 is pushed onto the stack: address of new node will be **1bb0**

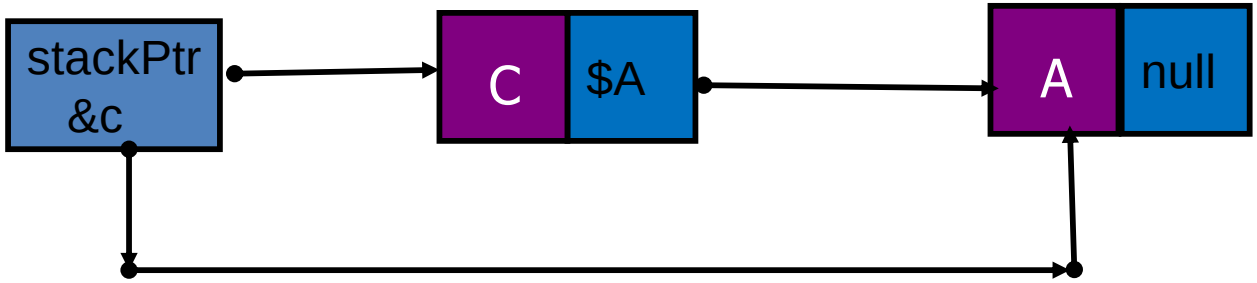


# Delete/pop a node from the stack

**The current stack containing two node**









- Assign the data element of (top/first) node in the stack to a variable Y
- Assign stackPtr pointer to point to second node in the stack.
- Return data and delete node ( c )

# Sample code pop function

denis.manley@apollo: ~/OS2/Week5_stack_queue

```
// remove a node from the stack top
int pop(StackNode* *topPtr)
{

   /* the error prevention code
    (ensuring the stack is not empty
     is in function main: if (stackPtr != NULL)
   */

   StackNode* tempPtr = *topPtr;
   int popValue = (*topPtr)->data;

   *topPtr = (*topPtr)->nextPtr; //move pointer to second node in stack

   free(tempPtr); // saftelyremove node


 return popValue;
}
```

121,1 74%

# Popping 2 nodes from stack: Stack2.c

```
The stack is:
12|0x55adcbc81ae0 (0x55adcbc81b00) --> 7|0x55adcbc81ac0 (0x55adcbc81ae0) --> 11|(nil) (0x55adcbc81ac0) --> NULL

Enter choice:
1 to push a value on the stack
2 to pop a value off the stack
3 to end program
? 2
****************** value of stack and frame pointers after the pop **********************

the value of the stack pointer is: 0x55adcbc81ae0
the value of the frame pointers is: 0x55adcbc81ac0
********************************************************************************

The popped value is 12.
The stack is:
7|0x55adcbc81ac0 (0x55adcbc81ae0) --> 11|(nil) (0x55adcbc81ac0) --> NULL

Enter choice:
1 to push a value on the stack
2 to pop a value off the stack
3 to end program
? 2
****************** value of stack and frame pointers after the pop **********************

the value of the stack pointer is: 0x55adcbc81ac0
the value of the frame pointers is: (nil)
********************************************************************************

The popped value is 7.
The stack is:
11|(nil) (0x55adcbc81ac0) --> NULL

Enter choice:
1 to push a value on the stack
2 to pop a value off the stack
3 to end program
```

# Print function:

```
// print the stack
void printStack(StackNode* currentPtr)
{
   // if stack is empty
   if (currentPtr == NULL) {
      puts("The stack is empty.\n");
   }
   else {
      puts("The stack is:");

      // while not the end of the stack
      while (currentPtr != NULL) {
         printf("%d | %p --> ", currentPtr->data,currentPtr->nextPtr);
         currentPtr = currentPtr->nextPtr;
      }

      puts("NULL\n");
   }
}
```

Why is currentPtr a “single” pointer to a node; What does it store?

# Home Work: Illustrate adding 3 nodes onto the stack

```
denis.manley@soc-apollo:~/OS2/Week5_stack_queue$ ./Stack2
the address of the stackPtr is 0x7ffcd1bcf3c0
Enter choice:
1 to push a value on the stack
2 to pop a value off the stack
3 to end program
? 1
Enter an integer: 10
****************** value of stack and frame pointers after the push ***********************

the value of the stack pointer is: 0x56184dc18ac0
the value of the frame pointers is: (nil)
********************************************************************************

The stack is:
10|(nil) (0x56184dc18ac0) --> NULL

Enter choice:
1 to push a value on the stack
2 to pop a value off the stack
3 to end program
? 1
Enter an integer: 25
****************** value of stack and frame pointers after the push ***********************

the value of the stack pointer is: 0x56184dc18ae0
the value of the frame pointers is: 0x56184dc18ac0
********************************************************************************

The stack is:
25|0x56184dc18ac0 (0x56184dc18ae0) --> 10|(nil) (0x56184dc18ac0) --> NULL

Enter choice:
1 to push a value on the stack
2 to pop a value off the stack
3 to end program
? 1
Enter an integer: 8
****************** value of stack and frame pointers after the push ***********************

the value of the stack pointer is: 0x56184dc18b00
the value of the frame pointers is: 0x56184dc18ae0
********************************************************************************

The stack is:
8|0x56184dc18ae0 (0x56184dc18b00) --> 25|0x56184dc18ac0 (0x56184dc18ae0) --> 10|(nil) (0x56184dc18ac0) --> NULL
```

What is the value in stackPtr after a pop?

# Home Work: The address after poping 2 nodes

```
The stack is:
8|0x56184dc18ae0 (0x56184dc18b00) --> 25|0x56184dc18ac0 (0x56184dc18ae0) --> 10|(nil) (0x56184dc18ac0) --> NULL

Enter choice:
1 to push a value on the stack
2 to pop a value off the stack
3 to end program
? 2
****************** value of stack and frame pointers after the pop **********************

the value of the stack pointer is: 0x56184dc18ae0
the value of the frame pointers is: 0x56184dc18ac0
********************************************************************************

The popped value is 8.
The stack is:
25|0x56184dc18ac0 (0x56184dc18ae0) --> 10|(nil) (0x56184dc18ac0) --> NULL

Enter choice:
1 to push a value on the stack
2 to pop a value off the stack
3 to end program
? 2
****************** value of stack and frame pointers after the pop **********************

the value of the stack pointer is: 0x56184dc18ac0
the value of the frame pointers is: (nil)
********************************************************************************

The popped value is 25.
The stack is:
10|(nil) (0x56184dc18ac0) --> NULL
```

# Sample Questions

- Describe, using an example, what is the stack data structure and how/where it can be used in the execution of a program . (6 marks)

- Describe, using examples, what is a queue data structure and give two examples of where it may used in operating systems. (6 marks)

# Sample question

- Explain, using examples (include node addresses), how to enqueue and dequeue nodes from of a queue **(15 marks)**

- Explain, using examples (include node addresses), how to push and pop functions of a stack **(15 marks)**

**Clearly explain, using an example, if this function will add a node on to the top of a stack : (you will need to explain all parts of the code)** (10 marks)

```
//the fnt  call     // StackPtr is a pointer to the top of the stack data structure.
Push (&StackPtr);

void push(StackNode** topPtr, int info)
{
   StackNode newPtr = malloc(sizeof(StackNode));


   if (newPtr != NULL) {
      newPtr->data = info;
      newPtr->nextPtr = topPtr;
      topPtr = newPtr;
   }
   else {
      printf("%d not inserted. No memory available.\n", info);
   }
}
```

Explain what the following code does and if it would correctly pop a value from the stack **(10 Marks)**

```
// function to pop values from stack assuming it is not empty
int pop(StackNode** topPtr)
{
   if (topPtr != Null){
   StackNode* tempPtr = topPtr;
   int popValue = (*topPtr)->data;
   topPtr = (*topPtr)->nextPtr;
   free(tempPtr);
   return popValue;
}
else return '\0';
```