

DYNAMIC MEMORY

File Input

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

```
int result, x, y;  
FILE *pFile;
```

```
pFile = fopen ("data/test.txt", "r");
```

```
result = fscanf (pFile, "%d %d", &x, &y);
```

```
fclose(pFile);
```

Dynamic Memory Allocation in C

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

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

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

Allocate an Array

```
int *pArray;  
const int SIZE = 1024;  
  
pArray = malloc(  
  
free(  

```

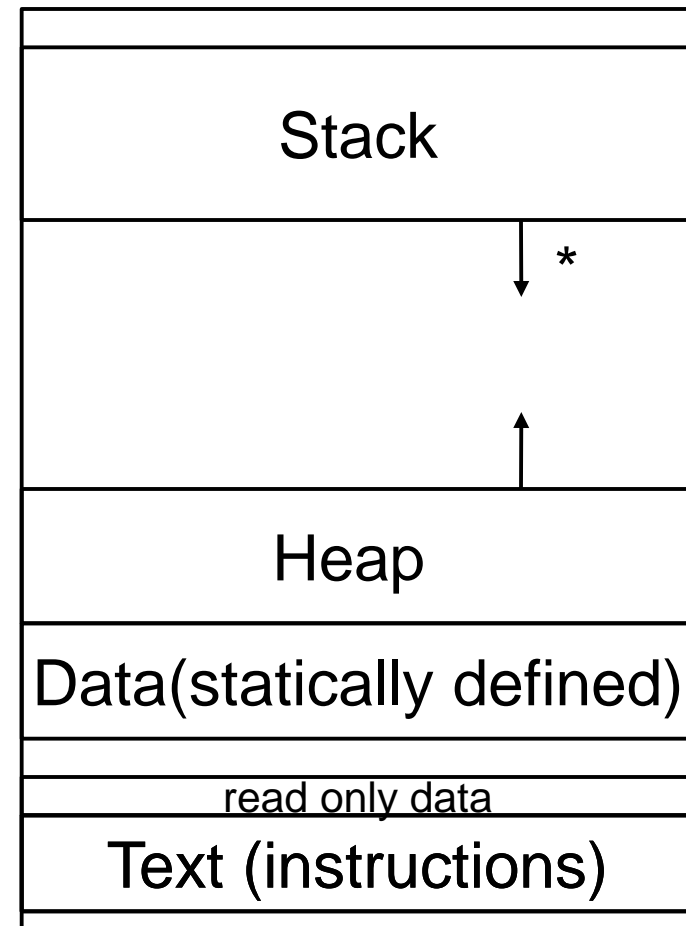
Memory Layout

- What is in each section?

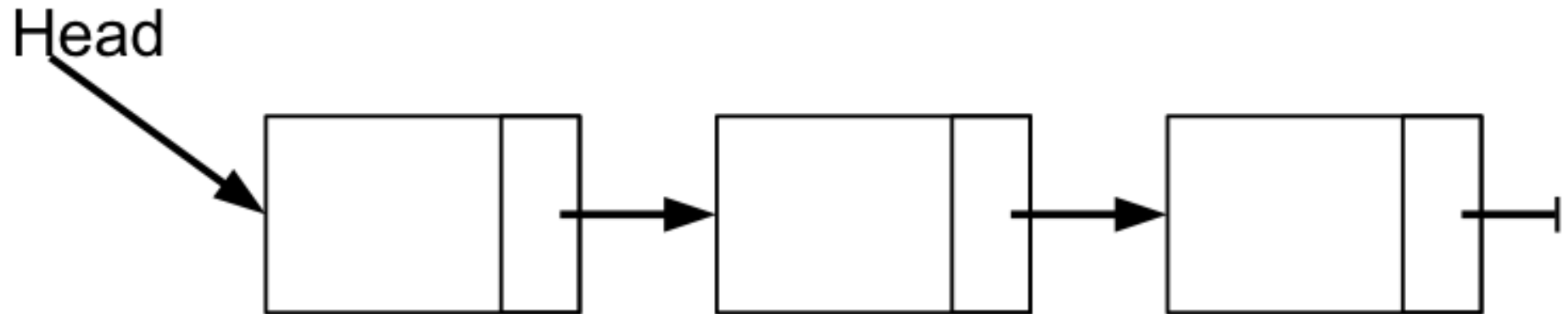
```
#include <stdio.h>
#include <stdlib.h>

int gValue = 9;
int gArray[1024];

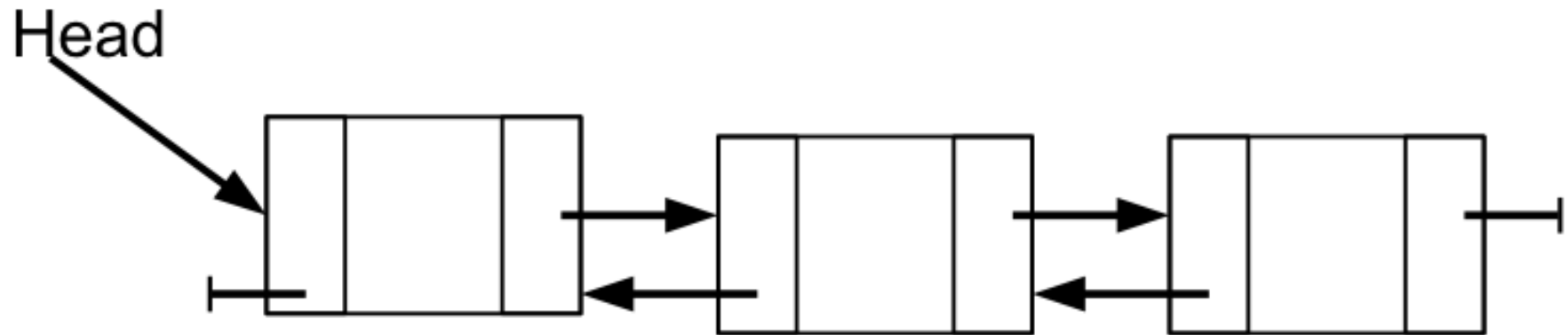
int main ()
{
    int *pArray;
    int value = 10;
    printf ("%d", gValue);
    pArray = (int*)
        malloc (sizeof (int) * 10)
    pArray[2] = 7;
    free (pArray);
    return 0;
}
```



Singly Linked List



Doubly Linked List



How to represent a node

```
typedef struct Node *NodePtr;  
typedef struct Node  
{  
    int data;  
    struct Node* psNext;  
} Node;  
  
Node sList;  
NodePtr psList;
```


Which of these are legal?

1. `sList.data = 5;`

2. `sList->psNext = NULL;`

3. `sList = NULL;`

4. `psList->data = 5;`

5. `psList = NULL;`

Problems

1. Create an empty list pointed to by **psList**.
2. Allocate space for a new node and set the list pointer to point to the new node.
3. Place the integer **10** into the data field of the single node.
4. Create another new node and place the integer **20** into the data field of the new node.
5. Link the two nodes together placing the node with 20 after the node 10.
6. A linked list exists pointed to by the list pointer **psList**. Write a function **length** that accepts the list pointer to a singly linked list and returns the length of the list.

Stack

```
typedef int DATATYPE;
typedef struct StackElement *StackElementPtr;
typedef struct StackElement
{
    DATATYPE data; // the user data
    StackElementPtr psNext;
} StackElement;

typedef struct Stack
{
    StackElementPtr psTop;
} Stack;
```

- `stkCreate(`
- `stkDispose(`
- `stkPush(`
- `stkPop(`
- `stkPeek(`