Struct

- A struct is a collection of variables under a single name. These variables can be of different data types.
- Sample code:

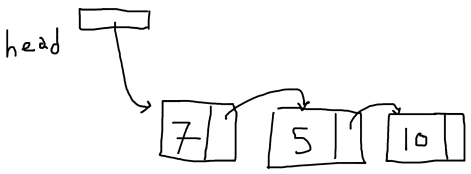
```
struct person{
    char name[10];
    int age;
};
int main(){
     struct person p1;
     struct person p2;
}
```

- This can be used to create advanced structures, such as Linked lists or Trees. typedef is used to make declaration easier. For example:

```
typedef struct node{
    int value;
    struct node *next;
}NODE;
int main(){
   NODE n;
}
```

- New nodes can be inserted in the linked list in the regular or reverse order. For example, the following function (additem) is insert new nodes at the beginning of the list:

```
void additem(NODE* h, int v){
   NODE* cell=(NODE*) malloc(sizeof(NODE));
   cell->value=v;
   cell->next=h;
   h=cell;
}
NODE* head;
additem(&head,10);
additem(&head,5);
additem(&head,7);
```



Multi-File Programs:

C programs can be divided into different files. In general, prototypes are saved in header files (.h). While functions' details should be defined in separate .c file(s). For example:

node.h

```
#include <stdio.h>
#include <stdlib.h>
typedef struct node{
   int value;
   struct node* next;
}NODE;
void additem(NODE* h, int v);
```

node.c

```
#include "node.h"
void additem(NODE* h, int v){
   NODE* cell=(NODE*) malloc(sizeof(NODE));
   cell->value=v;
   cell->next=h;
   h=cell;
}
```

```c
#include "node.h"
int main(){
   NODE* head;
   head = NULL;
   additem(&head,10);
   additem(&head,5);
   additem(&head,7);
   printf("%d\n",head->value);
}
```

program.c

- You will need to compile all .c files:

```
gcc node.c program.c
./a.out
```