Dynamic Memory in C

Dynamic Array Declaration

- Syntax:

```
array_type *array_name;
```

- Example (dynamic array of integers):

```
int * my_array;
```

- Initialization:

```
my_array = NULL;
```

Dynamic Structure Declaration

- Syntax:

```
struct struct_name {
      field_type1  field1;
      field_type2  field2;
      …
} *var1 = NULL;
typedef struct struct_name new_name;
```

- Note:
  - *var1* is optional, used to declare variable simultaneously w/ type
  - typedef is optional, used to simplify type name
  - Usage of constant **NULL** requires: **#include <stdlib.h>**

Dynamic Memory Allocation

- Syntax:

```
var = (type *)malloc(number_of_elements*sizeof(type));
```

- Note:

  malloc is type “void”, requiring typecasting to (*type **)

  sizeof is an integer function returning the size of a type in bytes

- Example (allocate memory for an array of 10 integers):

```
my_array = (int *)malloc(10*sizeof(int));
```

Dynamic Array/Structure Assignment

- Syntax:

```
var[index] = value;  /* array */
var[index].field = value; /*struct */
```

- Note:
  - *field* should be the same type as *value*
  - *index* is any value between 0 and "*number_of_elements*"-1 as defined in the malloc allocation