*"C programmers never die. They are just cast into void."*

*- Alan Perlis*

# CSE102
# Computer Programming with C

2020-2021 Spring Semester

## Dynamic Data Allocation Examples





1

```c
typedef struct engine {
    char name[20];
    int year;
    float volume;
    enum {v4=4, v5=5, v6=6, v8=8} cylinders;
} engine;
```

```c
void test() {
    engine e1 = {"Alfa Romeo", 1979, 2.5, v6}, e2, * e3;

    engine_print(e1);

    e2 = engine_read("engine1.txt");
    engine_print(e2);

    e3 = engine_read_dynamic("engine2.txt");
    engine_print(*e3);
}
```

More efficient?

```c
void engine_print(engine e) {
    printf("Engine: '%s' %d %.1fL v%d\n", e.name, e.year, e.volume, e.cylinders);
}
```



2

```c
typedef struct engine {
    char name[20];
    int year;
    float volume;
    enum {v4=4, v5=5, v6=6, v8=8} cylinders;
} engine;
```

```c
void test() {
    engine e1 = {"Alfa Romeo", 1979, 2.5, v6}, e2, * e3;

    engine_print(e1);

    e2 = engine_read("engine1.txt");
    engine_print(e2);

    e3 = engine_read_dynamic("engine2.txt");
    engine_print(*e3);
}
```

```
Ferrari F154
2014
2.9
8
```

```
McLaren M838T
2013
3.8
8
```

```c
engine engine_read(char * filename) {
    engine e;
    FILE * fin;
    fin = fopen(filename, "rt");
    fgets(e.name, 20, fin); e.name[strlen(e.name)-1] = 0;
    fscanf(fin, "%d %f %d", &(e.year), &(e.volume), &(e.cylinders));
    fclose(fin);
    return e;
}
```



3

```c
typedef struct engine {
    char name[20];
    int year;
    float volume;
    enum {v4=4, v5=5, v6=6, v8=8} cylinders;
} engine;
```

```c
void test() {
    engine e1 = {"Alfa Romeo", 1979, 2.5, v6}, e2, * e3;

    engine_print(e1);

    e2 = engine_read("engine1.txt");
    engine_print(e2);

    e3 = engine_read_dynamic("engine2.txt");
    engine_print(*e3);
}
```

```
Ferrari F154
2014
2.9
8
```

```
McLaren M838T
2013
3.8
8
```

```c
engine * engine_read_dynamic(char * filename) {
    engine e;
    FILE * fin;
    fin = fopen(filename, "rt");
    fgets(e.name, 20, fin); e.name[strlen(e.name)-1] = 0;
    fscanf(fin, "%d %f %d", &(e.year), &(e.volume), &(e.cylinders));
    fclose(fin);
    return &e;
}
```



4

```c
typedef struct engine {
    char name[20];
    int year;
    float volume;
    enum {v4=4, v5=5, v6=6, v8=8} cylinders;
} engine;
```

```c
void test() {
    engine e1 = {"Alfa Romeo", 1979, 2.5, v6}, e2, * e3;

    engine_print(e1);

    e2 = engine_read("engine1.txt");
    engine_print(e2);

    e3 = engine_read_dynamic("engine2.txt");
    engine_print(*e3);
}
```

```
Ferrari F154
2014
2.9
8
```

```
McLaren M838T
2013
3.8
8
```

```c
engine * engine_read_dynamic(char * filename) {
    engine * e;
    FILE * fin;
    fin = fopen(filename, "rt");
    e = (engine *)malloc(sizeof(engine));
    fgets(e->name, 20, fin); e->name[strlen(e->name)-1] = 0;
    fscanf(fin, "%d %f %d", &(e->year), &(e->volume), &(e->cylinders));
    fclose(fin);
    return e;
}
```



5


```c
typedef struct engine {
    char name[20];
    int year;
    float volume;
    enum {v4=4, v5=5, v6=6, v8=8} cylinders;
} engine;
```

```c
void test() {
    engine e1 = {"Alfa Romeo", 1979, 2.5, v6}, e2, * e3;

    engine_print(e1);

    e2 = engine_read("engine1.txt");
    engine_print(e2);

    e3 = engine_read_dynamic("engine2.txt");
    engine_print(*e3);
}
```




6


```c
typedef struct engine {
    char name[20];
    int year;
    float volume;
    enum {v4=4, v5=5, v6=6, v8=8} cylinders;
} engine;
```

```c
void test() {
    engine e1 = {"Alfa Romeo", 1979, 2.5, v6}, e2, * e3;

    engine_print(e1);

    e2 = engine_read("engine1.txt");
    engine_print(e2);

    e3 = engine_read_dynamic("engine2.txt");
    engine_print(*e3);
}
```


Let's make this dynamic – why?



7


```c
typedef struct engine {
    char name[20];
    int year;
    float volume;
    enum {v4=4, v5=5, v6=6, v8=8} cylinders;
} engine;
```

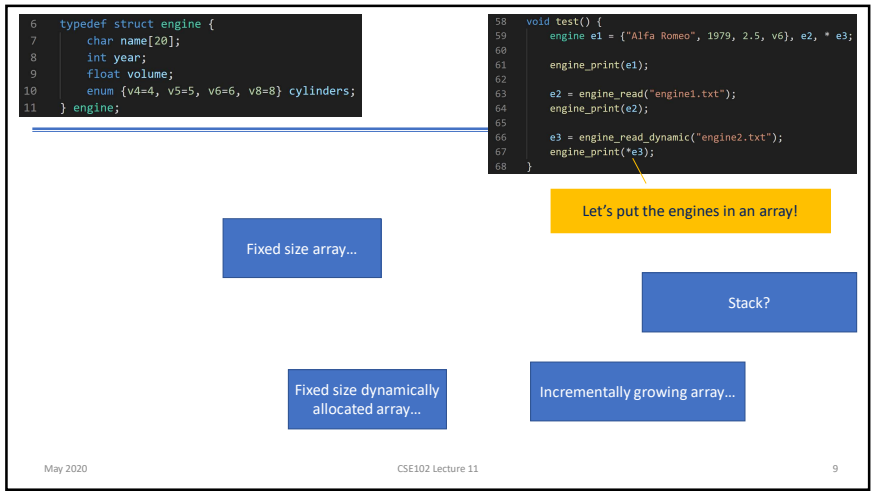
```c
void test() {
    engine e1 = {"Alfa Romeo", 1979, 2.5, v6}, e2, * e3;

    engine_print(e1);

    e2 = engine_read("engine1.txt");
    engine_print(e2);

    e3 = engine_read_dynamic("engine2.txt");
    engine_print(*e3);
}
```

Let's make this dynamic – why?

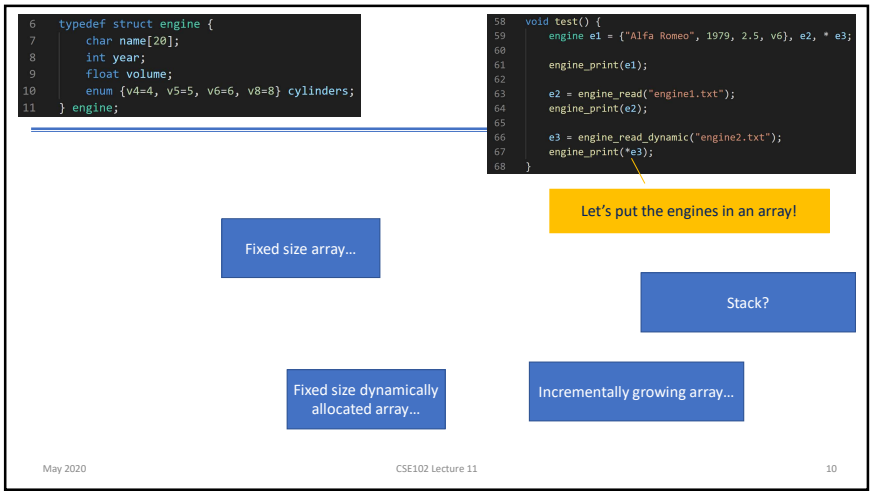
```c
engine * engine_read_dynamic(char * filename) {
    engine * e;
    FILE * fin;
    fin = fopen(filename, "rt");
    e = (engine *)malloc(sizeof(engine));
    fgets(e->name, 20, fin); e->name[strlen(e->name)-1] = 0;
    fscanf(fin, "%d %f %d", &(e->year), &(e->volume), &(e->cylinders));
    fclose(fin);
    return e;
}
```




8


6 typedef struct engine {
7 char name[20];
8 int year;
9 float volume;
10 enum {v4=4, v5=5, v6=6, v8=8} cylinders;
11 } engine;
58 void test() {
59 engine e1 = {"Alfa Romeo", 1979, 2.5, v6}, e2, * e3;
60
61 engine_print(e1);
62
63 e2 = engine_read("engine1.txt");
64 engine_print(e2);
65
66 e3 = engine_read_dynamic("engine2.txt");
67 engine_print(*e3);
68 }
Let's put the engines in an array!
Fixed size array…
Stack?
Fixed size dynamically allocated array…
Incrementally growing array…
May 2020
CSE102 Lecture 11
9

9


6 typedef struct engine {
7 char name[20];
8 int year;
9 float volume;
10 enum {v4=4, v5=5, v6=6, v8=8} cylinders;
11 } engine;
58 void test() {
59 engine e1 = {"Alfa Romeo", 1979, 2.5, v6}, e2, * e3;
60
61 engine_print(e1);
62
63 e2 = engine_read("engine1.txt");
64 engine_print(e2);
65
66 e3 = engine_read_dynamic("engine2.txt");
67 engine_print(*e3);
68 }
Let's put the engines in an array!
Fixed size array…
Stack?
Fixed size dynamically allocated array…
Incrementally growing array…
May 2020
CSE102 Lecture 11
10

10