Structures

Storing a date (02/25/22)

int month = 2, day = 25, year = 2015;

- Scenario 1: We need to store multiple dates
  - int month1, day2, year2;
- Scenario 2: Vaccination database needs both date and which dose
  - int month, day, year, doseNumber;
- These variables are logically related
- Better to group them together rather than accessing them separately.

Scenario 1: Storing multiple dates

2	25	2022	2	28	2022
0	1	2	3	4	5

Scenario 2: Storing elements of different types together

- Arrays are not capable of grouping elements of **different types** together

Structure for Storing Date

- Possible instead to define a structure called date
- Has three components: month, day, year
- The components of a structure is known as members

```
struct  date
{
    int  month;
    int  day;
    int  year;
};
```

structures as variable type

- Definition of the structure defines a new type
- Variables of type struct data can be declared like any other data type
  - Referred to as instances of the structure

```
/*Declaring a variable of type struct date */

struct date today;

/* Declaring multiple variables of type struct date */

struct date today, purchaseDate;
```

Dealing with structure variables

- Dealing with structure variables different from dealing with *int*, *float*, *double* etc.
- Accessing structure members
  - *variable_name*.*member_name* = *value*
  - No space permitted between variable name, period and member name

```
/* Declaring variable of type struct date */
struct date today

// Accessing structure members

today.month = 2;
today.day = 25;
today.year = 2022;
```

struct point

```c
struct point {
    double x;
    double y;
};

int main() {
    struct point point1;

    point1.x = 5.2;
    point1.y = -3.4;

    struct point point2 = {10.7, 2.8};

    return 0;
}
```

struct point

```c
struct point {
    double x;
    double y;
};

double distance(struct point p1, struct point p2) {
    double x_distance = p2.x - p1.x;
    double y_distance = p2.y - p1.y;

    return sqrt(x_distance * x_distance + y_distance * y_distance);
}
```

```c
int main() {
    struct point point1;

    point1.x = 5.2;
    point1.y = -3.4;

    struct point point2 = {10.7, 2.8};

    double dist = distance(point1, point2);

    printf("The distance between the two points is %lf\n", dist);

    return 0;
}
```

struct rectangle

```c
struct point {
    double x;
    double y;
};

struct rectangle {
    struct point lower_left;
    struct point upper_right;
};

double distance(struct point p1, struct point p2) {
    double x_distance = p2.x - p1.x;
    double y_distance = p2.y - p1.y;

    return sqrt(x_distance * x_distance + y_distance * y_distance);
}
```

```c
struct rectangle make_rectangle(struct point lower_left,
                                struct point upper_right) {
    struct rectangle rect;

    rect.lower_left = lower_left;
    rect.upper_right = upper_right;

    return rect;
}

void print_rectangle(struct rectangle rect) {
    printf("Lower left point: (%lf, %lf)\n", rect.lower_left.x,
           rect.lower_left.y);
    printf("Upper right point: (%lf, %lf)\n", rect.upper_right.x,
           rect.upper_right.y);
}
```

struct rectangle

```c
struct rectangle make_rectangle(struct point lower_left,
                                struct point upper_right) {
    struct rectangle rect;

    rect.lower_left = lower_left;
    rect.upper_right = upper_right;

    return rect;
}

void print_rectangle(struct rectangle rect) {
    printf("Lower left point: (%lf, %lf)\n", rect.lower_left.x,
           rect.lower_left.y);
    printf("Upper right point: (%lf, %lf)\n", rect.upper_right.x,
           rect.upper_right.y);
}
```

```c
int main() {
    struct point point1;

    point1.x = 5.2;
    point1.y = -3.4;

    struct point point2 = {10.7, 2.8};

    double dist = distance(point1, point2);

    printf("The distance between the two points is %lf\n", dist);

    struct rectangle rect = make_rectangle(point1, point2);

    print_rectangle(rect);

    return 0;
}
```