Code Examples from Lectures on 13-8_structs

COMP1911 22T2 (https://webcms3.cse.unsw.edu.au/COMP1911/22T2) Introduction to Programming (https://webcms3.cse.unsw.edu.au/COMP1911/22T2)

48.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/13-8_structs/code/48.c)

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

int main(int argc, char* argv[]) {

    char dylan[4] = {'0','1','2','3'};
    int i = 0;
    while (i < 4) {
        printf("%d\n", dylan[i] - '0');
        printf("%c\n", dylan[i]);
        i++;
    }
    return 0;
}
```

date.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/13-8_structs/code/date.c)

An example of defining a simple date struct and passing structs into a function and returning a struct from a function

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


typedef struct date Date;

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

// 3/5/2017
void printDate(Date d);
Date readDate(void);

int main(void){
    Date d = readDate();

    printDate(d);
    return EXIT_SUCCESS;
}

void printDate(Date d){
    printf("%d/%d/%d\n",d.day,d.month,d.year);
}

//3/5/2017
Date readDate(void){
    Date d;
    if(scanf("%d/%d/%d",&(d.day),&(d.month),&(d.year)) != 3){
       fprintf(stderr,"Error: date in wrong format\n");
       exit(EXIT_FAILURE);
    }
    return d;
}
```

studentDemo0.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/13-8_structs/code/studentDemo0.c)

A simple example of a defining a struct, initialising it with data and passing it into a function

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

#define MAX_LEN 100

struct student {
    int zid;
    double wam;
    char name [MAX_LEN];
    char address[MAX_LEN];
};

void printStudentDetails(struct student s);

int main(void){

    struct student s;
    struct student s2;
    s.zid = 1231412;
    s.wam = 89.5;
    strcpy(s.name, "Ronald McDonald");
    strcpy(s.address,"5 Fake Street");

    s2.zid = 321323;
    s2.wam = 10;
    strcpy(s2.name,"Jack Russell");
    strcpy(s2.address,"7 Fake Street");


    printStudentDetails(s);

    printStudentDetails(s2);


    return EXIT_SUCCESS;
}

//Move the print to a function
void printStudentDetails(struct student s){
    printf("z%d %s %lf %s\n",s.zid,s.name, s.wam,s.address);
}
```

dateArray.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/13-8_structs/code/dateArray.c)

Reads in and prints out up to 100 dates in the format of

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

#define MAX_DATES 100


typedef struct date Date;

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

// 3/5/2017
void printDate(Date d);
Date readDate(void);
int readDates(Date dates[],int maxDates);
void printDates(Date dates[], int numDates);


int main(void){
    Date importantDates[MAX_DATES];
    printf("Enter up to %d dates in the form d/m/y\n",MAX_DATES);
    int numDates = readDates(importantDates, MAX_DATES);
    printDates(importantDates,numDates);
    return EXIT_SUCCESS;
}

void printDate(Date d){
    printf("%d/%d/%d\n",d.day,d.month,d.year);
}

//3/5/2017
Date readDate(void){
    Date d;
    if(scanf("%d/%d/%d",&(d.day),
                        &(d.month),
                        &(d.year)) != 3){
       fprintf(stderr,"Error: date in wrong format\n");
       exit(EXIT_FAILURE);
    }
    return d;
}


//
//Index    0    1   2   3   4
//. day  |
//.month |
//.year  |
//We can't use the readDate function because it can't return
//the Date and the result of scanf
//We can do something like that by using pointers which
//we will see soon, but for now we read in
//each date in the array in this function
int readDates(Date dates[],int maxDates){
    int i = 0;
    while ( i < maxDates &&
            scanf("%d/%d/%d",&(dates[i].day),
                        &(dates[i].month),
                        &(dates[i].year)) == 3){

        i = i + 1;
    }
    return i;
}

void printDates(Date dates[], int numDates){
    int i = 0;
    while ( i < numDates ){
        printDate(dates[i]);
        i = i + 1;
    }
}
```

datePassByReference.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/13-8_structs/code/datePassByReference.c)

Show the use of passing dates by reference to allow reading in the date but also returning 1 or 0 to indicated whether the date was successfully read in.

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

#define MAX_DATES 100


typedef struct date Date;

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

// 3/5/2017
void printDate(Date d);
int readDate(Date * d);
int readDates(Date dates[],int maxDates);
void printDates(Date dates[], int numDates);


int main(void){
    Date d;
    //Read in one date
    if(readDate(&d) == 1){
        printDate(d);
    } else {
        fprintf(stderr,"That's not a date!\n");
        return EXIT_FAILURE;
    }

    //Read in lots of dates
    Date importantDates[MAX_DATES];
    int numDates = readDates(importantDates, MAX_DATES);
    printDates(importantDates,numDates);
    return EXIT_SUCCESS;
}

void printDate(Date d){
    printf("%d/%d/%d\n",d.day,d.month,d.year);
}

//Returns 1 if it successfully read in a date
//Returns 0 otherwise
//We need to use a pointer, so we can update the actual
//memory of the struct from the main
int readDate(Date * d){
    if ( scanf("%d/%d/%d",&(d->day),&(d->month),&(d->year)) == 3){
        return 1;
    } else {
        return 0;
    }
}

//
//Index    0    1   2   3   4
//. day  |
//.month |
//.year  |
//We can't use the readDate function because it can't return
//the Date and the result of scanf
//We can do something like that by using pointers which
//we will see soon, but for now we read in
//each date in the array in this function
int readDates(Date dates[],int maxDates){
    int i = 0;
    while ( i < maxDates && readDate(&dates[i]) ==1){

        i = i + 1;
    }
    return i;
}

void printDates(Date dates[], int numDates){
    int i = 0;
    while ( i < numDates ){
        printDate(dates[i]);
        i = i + 1;
    }
}
```

doublemal.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/13-8_structs/code/doublemal.c)

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

#define SIZE 10

int main(int argc, char* argv[]) {

    int i = 5;
    int *p = &i;
    free(p);

    printf("i = %d\n", i);
    printf("&i = %p\n", &i);
    printf("p = %p\n", p);
    printf("*p = %d\n", *p);
    return 0;

    double *dylan;
    dylan = malloc(sizeof(double) * SIZE);
    free(dylan);
    dylan = NULL;

    dylan = malloc(sizeof(double) * SIZE);
    free(dylan);
    dylan = NULL;

    dylan = malloc(sizeof(double) * SIZE);
    free(dylan);
    dylan = NULL;

    dylan = malloc(sizeof(double) * SIZE);
    free(dylan);
    dylan = NULL;

    dylan = malloc(sizeof(double) * SIZE);
    free(dylan);
    dylan = malloc(sizeof(double) * SIZE);
    free(dylan);
    dylan = malloc(sizeof(double) * SIZE);
    free(dylan);
    dylan = NULL;

    printf("%p\n", dylan);
    return 0;
}
```

mallocStudentArray.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/13-8_structs/code/mallocStudentArray.c)

Using malloc to create our own array on the heap instead of the stack. This means we can return the array from a function

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

#define MAX 100
#define MAX_LEN 25

typedef struct person Person;
struct person {
    double mark;
    char name[MAX_LEN];

};

Person *  readMarks(int size);
void printMarks(Person  marks[], int size);

int main(int argc, char * argv[]){
   Person * marks;
   int numMarks;

   printf("How many marks are you entering: ");
   scanf("%d",&numMarks);


   //Enter in format like
   // 99.5 Kelly
   // 70.3 Jack
   // 0.5 Fido
   printf("Please enter your marks : \n");
   marks = readMarks(numMarks);

   printf("Your marks are: ");
   printMarks(marks,numMarks);

   free(marks);
   return EXIT_SUCCESS;
}

Person * readMarks ( int size ){

   Person * marks = malloc(sizeof(struct person)*size);

   if (marks == NULL) {
        fprintf(stderr,"Malloc failed...no more memory\n");
        exit(EXIT_FAILURE);
   }
   int i = 0;
   while(i < size){

      scanf("%lf",&(marks[i].mark));
      fgets(marks[i].name,MAX_LEN,stdin);
      i++;
   }

   return marks;
}

void printMarks(Person marks[], int size){
   int i = 0;
   while(i < size){
      printf("%lf %s",marks[i].mark,marks[i].name);
      i++;
   }
   printf("\n");
}
```

memoryAddress.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/13-8_structs/code/memoryAddress.c)

Demonstrating the address of & operator and size of with a struct

Showing where variables are stored in relation to each other

For best results use gcc -o memoryAddress memoryAddress.c

./memoryAddress

```c
#include <stdio.h>

typedef struct date Date;

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

int main(void) {

    Date d;

    //Address of d is the same as the address of d.day
    //These are all next to each other in memory
    printf("Address of d is %p\n",&d);
    printf("Address of d.day is %p\n",&d.day);
    printf("Address of d.month is %p\n",&d.month);
    printf("Address of d.year is %p\n",&d.year);

    printf("Size of date is %d\n",sizeof(struct date));


    return 0;
}
```

nestedStudent0.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/13-8_structs/code/nestedStudent0.c)

A simple example of a defining nested struct

Each student has a field called dob, which is also a struct of type Date

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

#define MAX_LEN 100

typedef struct date Date;

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

typedef struct student Student;

struct student {
    int zid;
    double wam;
    Date dob;
    char name [MAX_LEN];

};

void printStudent(Student s);

int main(void){

    Student s;
    s.zid = 1231412;
    s.wam = 89.5;
    //the dob field is inside the student, s
    //and the day field is inside the dob field
    s.dob.day = 8;
    s.dob.month = 5;
    s.dob.year = 2017;
    strcpy(s.name, "Ronald McDonald");

    printStudent(s);

    return EXIT_SUCCESS;
}

void printStudent(Student s){
    printf("z%d %s %d/%d/%d %lf\n",s.zid,s.name,s.dob.day,s.dob.month,s.dob.year, s.wam);
}
```

nestedStudent1.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/13-8_structs/code/nestedStudent1.c)

A simple example of a defining nested struct

Each student has a field called dob, which is also a struct of type Date

In this version we have created some functions to read in data and have broken down our printStudent function to use a printDate function.

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

#define MAX_LEN 100

typedef struct date Date;

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

typedef struct student Student;

struct student {
    int zid;
    double wam;
    Date dob;
    char name [MAX_LEN];

};


void printDate(Date d);
void printStudent(Student s);
int readDate(Date * d);
int readStudent(Student *s);

int main(void){

    Student s;
    if(readStudent(&s)){
        printStudent(s);
    }
    return EXIT_SUCCESS;
}


//Returns 1 if it successfully read in a date and 0 otherwise
int readDate(Date * d){
    if( scanf("%d %d %d",&d->day,&d->month,&d->year) == 3){
        return 1;
    } else {
        return 0;
    }
}

//Returns 1 if it successfully read in a student and 0 otherwise
int readStudent(Student *s){
    if((scanf("%d %lf",&s->zid,&s->wam) == 2) && (readDate(&s->dob) != 0) &&
       fgets(s->name,MAX_LEN,stdin) != NULL){
         //Remove trailing new line character
         int lenName = strlen(s->name);
         s->name[lenName -1] = '\0';
         return 1;
    }
    return 0;
}


void printDate(Date d){
    printf("%d/%d/%d",d.day,d.month,d.year);
}

void printStudent(Student s){
    printf("z%d %s ",s.zid,s.name);
    printDate(s.dob);
    printf(" %lf\n",s.wam);
}
```

nestedStudentArray.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/13-8_structs/code/nestedStudentArray.c)

An example of and array of nested structs

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

#define MAX_STUDENTS 100
#define MAX_LEN 100

typedef struct date Date;

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

typedef struct student Student;

struct student {
    int zid;
    double wam;
    Date dob;
    char name [MAX_LEN];

};


void printDate(Date d);
void printStudent(Student s);
int readDate(Date * d);
int readStudent(Student *s);
int readAllStudents(Student s[], int maxSize);
void printAllStudents(Student s[], int size);

int main(void){

    Student s[MAX_STUDENTS];
    int numStudents = readAllStudents(s,MAX_STUDENTS);
    printAllStudents(s,numStudents);
    return EXIT_SUCCESS;
}


//Returns 1 if it successfully read in a date and 0 otherwise
int readDate(Date * d){
    if( scanf("%d %d %d",&d->day,&d->month,&d->year) == 3){
        return 1;
    } else {
        return 0;
    }
}

//Returns 1 if it successfully read in a student and 0 otherwise
int readStudent(Student *s){
    if((scanf("%d %lf",&s->zid,&s->wam) == 2) && (readDate(&s->dob) != 0) &&
       fgets(s->name,MAX_LEN,stdin) != NULL){
         //Remove trailing new line character
         int lenName = strlen(s->name);
         s->name[lenName -1] = '\0';
         return 1;
    }
    return 0;
}


void printDate(Date d){
    printf("%d/%d/%d",d.day,d.month,d.year);
}

void printStudent(Student s){
    printf("z%d %s ",s.zid,s.name);
    printDate(s.dob);
    printf(" %lf\n",s.wam);
}

void printAllStudents(Student s[], int size){
    for(int i =0; i < size; i++){
        printStudent(s[i]);
    }
}

//return the number of students read in
int readAllStudents(Student s[], int maxSize){
    int i = 0;
    while( i < maxSize && readStudent(&s[i]) ){
```

```
        i++;
    }
    return i;
}
```

null.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/13-8_structs/code/null.c)

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

int main(int argc, char* argv[]) {

    int *s = NULL;
    printf("%d\n", *s);
    return 0;
}
```

stringmal.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/13-8_structs/code/stringmal.c)

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

// string char*

int main(int argc, char* argv[]) {

    int size;

    printf("Please enter size of string: ");
    scanf("%d", &size);

    char name[size + 1];

    printf("Please enter your string: ");
    scanf("%s", name);
    printf("%d\n", size);
    printf("%s\n", name);
    return 0;
}
```

stringmal2.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/13-8_structs/code/stringmal2.c)

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

// string char*

/*char *conversion(int power) {
    char *s = NULL;
    if (power == 1) {
        s = "0x2";
    } else if (power == 2) {
        s = "0x4";
    }
    return s;
}*/

int main(int argc, char* argv[]) {

    int size;
    char *name;
    printf("Please enter size of string: ");
    scanf("%d", &size);

    //char name[size + 1];
    name = malloc(sizeof(char) * (size + 1));

    printf("Please enter your string: ");
    scanf("%s", name);
    printf("%d\n", size);
    printf("%s\n", name);

    free(name);
    name = NULL;
    return 0;
}
```

struct.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/13-8_structs/code/struct.c)

```c
#include <stdio.h>

#define MAX_NAME 64
#define N_LABS 9

struct student {
    int zid; // int
    char *name; // char *
    double labMarks[N_LABS]; // double *
    double ass1Mark; // double
    double ass2Mark; // double
};
typedef struct student Student;

void takeMyStruct(Student thisStudent) {
    // do stuff with it
}

int main(int argc, char* argv[]) {

    printf("int: %lu\n", sizeof(int));
    printf("int: %lu\n", sizeof(double[N_LABS]));
    printf("int: %lu\n", sizeof(Student));

    Student littleHayden;
    littleHayden.zid = 1234567;
    littleHayden.name = "Little Hayden";
    littleHayden.labMarks[0] = 0;
    littleHayden.labMarks[1] = 0;
    littleHayden.labMarks[2] = 0;
    littleHayden.labMarks[3] = 0;
    // ...
    littleHayden.ass1Mark = 4;
    littleHayden.ass2Mark = 0;

    takeMyStruct(littleHayden);

    return 0;
}
```

structInit.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/13-8_structs/code/structInit.c)

An example of using initialisers for structs

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

#define MAX_LEN 100

typedef struct date Date;

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

typedef struct student Student;
struct student {
    int zid;
    double wam;
    Date dob;
    char name [MAX_LEN];
};

void printStudent(Student s);
void printStudents(Student students[], int n);
int main(void){

    Student s = {1231412,89.5,{16,5,2018},"Ronald McDonald"};
    printStudent(s);
    int nums[100] = {99};
    Student students[100] = {{1231412,89.5,{16,5,2018},"Ronald McDonald"}};
    printStudents(students,100);

    return EXIT_SUCCESS;
}

void printStudents(Student students[], int n){
    int i = 0;
    while ( i < n ){
        printStudent(students[i]);
        i = i + 1;
    }
}

void printDate(Date d){
    printf("%d/%d/%d",d.day,d.month,d.year);
}

void printStudent(Student s){
    printf("z%d %s ",s.zid,s.name);
    printDate(s.dob);
    printf(" %lf\n",s.wam);
}
```

studentDemo1.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/13-8_structs/code/studentDemo1.c)

A simple example of a defining a struct, initialising it with data and passing it into a function

This time we have defined a typedef so we can use Student instead of having to type struct student everywhere

Compare to studentDemo0.c

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

#define MAX_LEN 100


typedef struct student Student;

#define MAX_LEN 100

struct student {
    int zid;
    double wam;
    char name [MAX_LEN];
    char address[MAX_LEN];
};

void printStudentDetails(Student s);

int main(void){

    Student s;
    Student s2;
    s.zid = 1231412;
    s.wam = 89.5;
    strcpy(s.name, "Ronald McDonald");
    strcpy(s.address,"5 Fake Street");

    s2.zid = 321323;
    s2.wam = 10;
    strcpy(s2.name,"Jack Russell");
    strcpy(s2.address,"7 Fake Street");


    printStudentDetails(s);

    printStudentDetails(s2);


    return EXIT_SUCCESS;
}

//Move the print to a function
void printStudentDetails(Student s){
    printf("z%d %s %lf %s\n",s.zid,s.name, s.wam,s.address);
}
```

studentDemo2.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/13-8_structs/code/studentDemo2.c)

An example of the properties of structs

We can use = to get a copy of a struct

When we pass a struct into a function we pass a copy

We can't compare whole structs using == we need to compare the fields inside ourselves

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

#define MAX_LEN 100

typedef struct student Student;

struct student {
    int zid;
    double wam;
    char name [MAX_LEN];
    char address[MAX_LEN];
};

//1 equal
//0 not
int compareStudents(Student s1, Student s2);

void printStudentDetails(Student s);

int main(void){
    Student s;
    Student s2;
    s.zid = 1231412;
    s.wam = 89.5;
    strcpy(s.name, "Ronald McDonald");
    strcpy(s.address,"5 Fake Street");

    printStudentDetails(s);
    //This should print exactly the same details again
    //since changes made to the wam in the printStudentDetails function
    //are only acting on a copy
    printStudentDetails(s);


    // We can use = and get a copy of a struct
    s2 = s;

    printStudentDetails(s);
    printStudentDetails(s2);

    // We can't use == for structs
    // if ( s == s2){
    if ( compareStudents(s,s2) == 1 ){
        printf("s equals s2\n");
    } else {
        printf("NOT THE SAME\n");
    }

    //s2 is a copy of s, so changes to s2 won't affect s
    s2.zid = 6666666;
    s2.wam = 100;

    printStudentDetails(s);
    printStudentDetails(s2);

    if ( compareStudents(s,s2) == 1 ){
        printf("s equals s2\n");
    } else {
        printf("s does not equal s2\n");
    }

    return EXIT_SUCCESS;
}

// structs are passed by copy
void printStudentDetails(Student s){
    printf("z%d %.2lf %s %s\n",s.zid,s.wam,s.name,s.address);
    s.wam = 0; //THIS WON"T CHANGE THE struct back in main
               //since structs are passed by copy

}

// This function returns 1 if contents of the structs
// are equal and 0 otherwise.
int compareStudents(Student s1, Student s2){
   if(s1.zid == s2.zid && s1.wam == s2.wam &&
      strcmp(s1.name,s2.name) == 0 &&
      strcmp(s1.address,s2.address) == 0){
          return 1;
       } else {
          return 0;
    }
```

```
}
```

studentDemoArray.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/13-8_structs/code/studentDemoArray.c)

An example with an array of students

We demonstrate that passing an array of structs is passing by reference.

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

#define MAX_LEN 100

#define MAX_STUDENTS 400

typedef struct student Student;

struct student {
    int zid;
    double wam;
    char name [MAX_LEN];
    char address[MAX_LEN];
};


void printAllDetails(Student students[], int n);
void printStudentDetails(Student s);

int main(void){

    Student students[MAX_STUDENTS];

    students[0].zid = 1323234;
    students[0].wam =89.5;
    strcpy(students[0].name,"Ronald Mc Donald");
    strcpy(students[0].address,"5 Fake Street");


    students[1].zid = 321323;
    students[1].wam = 10;
    strcpy(students[1].name,"Jack Russell");
    strcpy(students[1].address,"7 Fake Street");

    printAllDetails(students,2);
    //The changes we made to students[0].wam inside the function
    //will have changed our struct in the main, since we
    //passed in the whole array (by reference).
    printAllDetails(students,2);

    return EXIT_SUCCESS;
}

void printStudentDetails(Student s){
   printf("z%d %.2lf %s %s\n",s.zid,s.wam,s.name,s.address);
   s.wam = 0; //THIS WON"T CHANGE THE struct back in main
               //since structs are passed by copy
}

void printAllDetails2(Student s[], int size){
    printf("\nPrinting all students\n");
    int i =0;
    while (i < size){
        printf("z%d %s %lf %s\n",
               s[i].zid, s[i].name, s[i].wam,s[i].address);
        i = i + 1;
    }
    //Since arrays are passed by reference, we are changing the struct
    //inside the array back in the main!
    s[0].wam = 0;
}

//This function is better than printAllDetails2 as we are using our
//existing printStudentDetails function rather than duplicating code
void printAllDetails(Student students[], int n){
    printf("\nPrinting all students\n");
    int i = 0;
    while ( i < n ){
        printStudentDetails(students[i]);
        i = i + 1;
    }
    //Since arrays are passed by reference, we are changing the struct
    //inside the array back in the main!
    students[0].wam = 0;
}
```

toby.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/13-8_structs/code/toby.c)

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

#define NAME_MAX_SIZE 100
#define ARRAY_SIZE 3

struct human {
    int weight;
    int height;
    int age;
    int shoesize;
    char name[NAME_MAX_SIZE];
};

typedef struct human Human;

int BMI(Human h) {
    return h.weight * h.height * h.shoesize;
}

int main(int argc, char* argv[]) {

    Human person[ARRAY_SIZE];
    int i = 0;
    while (i < ARRAY_SIZE) {
        printf("What is their name? ");
        scanf("%s", person[i].name);

        printf("What is their weight? ");
        scanf("%d", &person[i].weight);

        printf("What is their height? ");
        scanf("%d", &person[i].height);

        printf("What is their age? ");
        scanf("%d", &person[i].age);

        printf("What is their shoe size? ");
        scanf("%d", &person[i].shoesize);

        i++;
    }
    i = 0;
    while (i < ARRAY_SIZE) {
        printf("%s's BMI is %d\n", person[i].name, BMI(person[i]));
        i++;
    }

    return 0;
}
```

typedef.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/13-8_structs/code/typedef.c)

```c
#include <stdio.h>

#define NUMBER 4

typedef double real;
typedef double real2;
typedef double real3;
typedef double real4;
typedef real * realPointer;

int main(int argc, char* argv[]) {
    double a = NUMBER;
    double *aPointer = &a;

    real b = NUMBER;
    realPointer bPointer = &b;

    real c = NUMBER + 1;
    realPointer cPointer = &c;

    return 0;
}
```