The University Of Sheffield.

ACS126 Computing and Systems Design

Lecture 14. Dynamic memory allocation. Structures.

Department of Automatic Control
and Systems Engineering

The University Of Sheffield.

Module roadmap

C labs

1	2	3	4	5	6	7	8	9	10	11	12

Arrays and Strings, Pointers, Modular program design, Functions and passing parameters

Robot software due

Arrays and pointers, Multi-dimensional arrays, Dynamic memory, allocation Structures, Arrays of structures

Switch-case, Recursion, Putting it all together – using tools for problem solving.

C program due

Module review and exam pre-paration

C program due

Exam

Department of Automatic Control and Systems Engineering

The University Of Sheffield.

Learning Outcomes

By the end of today's session, you should be able to:

- Dynamically allocate memory for an array
- Define, declare and implement structures in C

The University Of Sheffield.

Dynamic Allocation of Memory for an Array

- What if you do not know how big an array needs to be? You have two choices:
  1. Estimate a maximum size and ensure this isn't exceeded in the program.
  2. Allocate memory for the array dynamically, i.e. while the program is executing.

The University Of Sheffield.

calloc

- Use `calloc` to get the right amount of storage for the array. `calloc` is in `stdlib.h`.
- Declare a pointer variable, of the correct type for the array.
- The parameters to calloc are:
  - the number of elements needed and
  - the **size of each element**.
- The return value from calloc is a pointer to void, which should be **cast** as a pointer to the correct type, so that it can be dereferenced later.

E.g.

```
int *listPtr;
listPtr = (int *)calloc(200,sizeof(int));
```

Declare a pointer

Number

Size

Cast as a pointer

Things you need, for dynamic allocation of memory for an array

- **Size of each element**: use `sizeof`: an operator that returns the number of bytes of storage allocated on that particular platform for that type of variable.

E.g. `sizeof(int)` `sizeof(float)`

- **Cast**: when a type is placed in parentheses before a variable on the right hand side of an assignment statement, the variable is first fetched from memory, then it is changed to the cast type for use in the assignment. The type of the variable in memory is not changed.

E.g.
```
float x;
int y;
y = (int)x;
```

Note: casting a float as an integer will mean that the storage as floating point will be ignored and the value will be treated as though stored as an integer.

The University Of Sheffield.

Freeing dynamically allocated memory

- After you have finished with a dynamically allocated block of memory, it is good practice to free it. You could then reuse this memory. If your program does not free the memory, it will be freed when the program terminates.

- E.g.

```
free(listPtr);
```

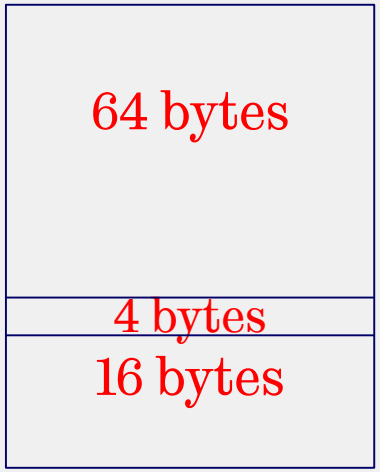
Complete Exercise

```
void printArray(int listSize, int array[]);
main()
{       int *listPtr, listSize, sizeInt, i;
      sizeInt = sizeof(int);
      printf("how big do you want the list to be?\n");
      scanf("%d",&listSize);
      listPtr = (int*)calloc(listSize,sizeInt);
      printArray(listSize,listPtr);
      free(listPtr); }
void printArray(int listSize, int array[]){
int i;
for(i=0; i<listSize; i++)
{   array[i]=i*10;
    printf("array contents at location %d is %d\n",i,
array[i]);     }  }
```

Department of Automatic Control
and Systems Engineering

The University Of Sheffield.

Storage template for student structure



Name

regNumber

yearlyWeightedMeans

STRUCTURES IN C

Defining Structures

Declaring Structures

Using structures

Department of Automatic Control
and Systems Engineering

The University Of Sheffield.

What is a Structure in C?

- A “derived” data type.
- Defining a structure in a program creates a new data type for use in that program.
- Useful for clustering information that is related in some way.

The University Of Sheffield.

Members of a Structure

- The members can be:
  - Variables (of types int, float, double, char)
  - Pointers to variables
  - Arrays (of types int, float, double, char)
  - Other structures

The University Of Sheffield.

Defining a Structure in C?

This gives you a new **type** called "student".

E.g.

```
struct student
{
    char Name[64];
    int regNumber;
    float yearlyWeightedMeans[4];
};
```

It has members Name, regNumber, and yearlyWeightedMeans.

The University Of Sheffield.

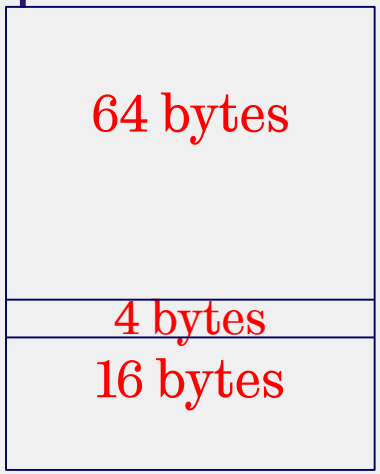
Structures in C

- After any #includes and #defines you have in your program
- Before main( ) and function definitions

What happens when you define a structure in a C program?

- The compiler creates a **template** for the storage of the data items you have specified in the structure.
- E.g.

Storage template for student structure



Name

regNumber

yearlyWeightedMeans

Department of Automatic Control and Systems Engineering

The University Of Sheffield.

Declaring a `struct` type

- If you have **defined** a structure in your program, you can then **declare** variables of that type.

E.g.

```
struct student
   {   char Name[64];
       int regNumber;
       float yearlyWeightedMeans[4];};

main()
{
    struct student myStudent;
                 ….
}
```

This declares a variable "myStudent", of type student.

Using Members of a Structure

- A structure has been defined (outside main), and declared in main.
- How do we use the member of a structure?
  - `myStudent.Name`
  - `myStudent.regNumber`
  - `myStudent.yearlyWeightedMeans[i]`

Variable Name . Data member

The University Of Sheffield.

Example struct

```
struct student   // the name of the struct is student
{
  char Name[64];   // these are the members of the struct
  int regNumber;
  float yearlyWeightedMeans[4];
};       // don't forget to put a semicolon after a struct definition
main(){
  struct student myStudent;   // declare struct of type student
  int i;
// you can read values into the members of the struct
  printf("please enter the students name (max 64
chars): ");
  gets(myStudent.Name);
```

```
  printf("\n Enter the regNumber of the student (an
integer): ");
  scanf("%d", &myStudent.regNumber);
  for(i=0; i<4; i++)  {
    printf("Enter the weighted means for year %d: ",
i+1);
    scanf("%f", &myStudent.yearlyWeightedMeans[i]);
  }
 printf("the student's name is %s\n", myStudent.Name);
  printf("the student's reg number is %d\n",
myStudent.regNumber);
     printf("the Student's yearly averages were as
follows:");
  for(i=0; i<4; i++)  {
    printf("Year %d mean is %0.2f", i+1,
   myStudent.yearlyWeightedMean[i]);
  }
```

Exercise

- Explain the code on your sheet. 5 minutes!

Passing Structures to Functions

```
struct fruit
{       char type[25];
        float costPerFruit;
};
void enterFruitData(struct fruit *F);
main()
{
        struct fruit myFruit;
        enterFruitData(&myFruit);
printf("Type of fruit: %s Cost per fruit: %.2f\n",
myFruit.type,myFruit.costPerFruit);
}
void enterFruitData(struct fruit *F)
{
       strcpy((*F).type,"apple");
        (*F).costPerFruit=0.25;
}
```

Dereference the pointer

Department of Automatic Control
and Systems Engineering

The University Of Sheffield.

Learning Outcomes

By the end of today's session, you should be able to:

- Dynamically allocate memory for an array
  - Using calloc to allocate space dynamically for an array, when you don't know how much space it will need before you run the program.
- Define, declare and implement structures in C
  - Use keyword `struct` to declare a structure with various data members
  - Use variableName.DataMember to manipulate the data members.

The University Of Sheffield.

Looking Ahead

C structures foreshadow C++ objects/classes

Department of Automatic Control and Systems Engineering