

Brief Intro to C (II)

Array

- Can use stack memory or heap memory.
 - stack memory: `int a [5];`
 - access array elements by using array index: `a[i]`
 - heap memory: `int *b = malloc(5 * sizeof(int));`
 - access array element by using pointer: `*(b+i)`

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

int main () {
    int i;

    // Way 1: create in stack
    int a[5];

    for (i = 0; i < 5; i++) {
        a[i] = i * 10;
    }

    for (i = 0; i < 5; i++) {
        printf("%d ", a[i]);
    }

    printf("\n");

    // Way 2: create in heap
    int *b;
    b = (int *)malloc(5 * sizeof(int));

    for (i = 0; i < 5; i++) {
        *(b+i) = i * 100;
    }

    for (i = 0; i < 5; i++) {
        printf("%d ", *(b+i));
    }

    free(b);

    printf("\n");

    return 0;
}
```

String

- There is no type “string” in C.
- To represent a string, we use a char array.
- Strings in C terminates with “\0”, so the size of a string is one more than the number of characters the string has.

```
#include <stdio.h>

int main (int argc, char *argv[]) {
    char string[] = "Hello world!";

    int i = sizeof(string)/sizeof(char);
    printf("size of the string: %d\n", i);

    for (int j = 0; string[j] != '\0'; j++) {
        printf("%d ", string[j]);
    }
    printf("\n");
}
```

- `string.h` provides more functions for strings
 - **int strcmp(const *str1, const *str2)**
 - < 0: the ascii value of the first different char in str1 is less than str2
 - = 0: strs are the same
 - > 0: the ascii value of the first different char in str1 is larger than str2
 - **char* strcpy(char *dest, const char* src)**
 - copy the src to the dest
 - to avoid overflow, the *size of dest should be large enough to contain the whole src string including the terminator NULL*

```
/**
 * Copy a string using strcpy
 * stringcpy2.c
 */

#include <stdio.h>
#include <string.h>

int main (int argc, char *argv[]) {
    char string1[] = "Hello world!";
    char string2[20];

    strcpy(string2, string1);
    printf("%s\n", string2);

    return 0;
}

/**
 * Compare two string using strcmp
 * stringcmp.c
 */

#include <stdio.h>
#include <string.h>

int main (int argc, char *argv[]) {
    char str1[] = "this is";
    char str2[] = "a test.";

    if (strcmp(str1, str2) == 0) {
        printf("Strings are equal. \n");
    }
    else {
        printf("Strings are different. \n");
    }

    return 0;
}
```

Struct

- like the class in java, but without member functions
- it is a data type that can enclose multiple data types
- way to define a struct

```
struct Tag { // Tag is the name of the struct
    member 1; // members can be basic data types or other
              structs
    member 2;
    ...
};
```

- to access a member, use .

```
/**
 * Example of structure
 * structexp.c
 */

#include <stdio.h>
#include <string.h>

struct people {
    char name[25];
    char gender; //F or M
    char bday[8]; //YYYYMMDD, this is a bug, size should be 9, \0
    char address[50];
};

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

    struct people person;

    /*initialize student1*/
    strcpy(person.name, "Sherlock Holmes");
    person.gender = 'M';
    strcpy(person.bday, "18540106");
    strcpy(person.address, "221B Baker Street, London");

    /*print student1 info*/
    printf("%s: %c, %s, %s\n",
           person.name,
           person.gender,
           person.bday,
           person.address);

    return 0;
}
```

- What is the bug in this program? `bday[8]` should be `bday[9]`

Typedef

- used to create a synonym for an existing data type
- If a struct is defined a synonym using typedef, we can use the synonym to declare an instance without using the keyword struct.
- If the instance is a pointer, we need to use -> to access the member of the struct instance.

```
/**
 * typedef example
 * typedef is used to create a synonym for an existing data type
 * synonym.c
 */

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

typedef unsigned int size_i;

typedef struct Mypuppy {
    char name[100]; // changing 100 to 99 won't impact the struct size why?
    [padding]
    int year;
    float weight;
} dog;

int main () {
    size_i a = 1;

    printf("size of a is %lu, value is %d\n", sizeof(size_i), a);
    printf("size of dog: %lu\n", sizeof(dog));

    dog* toto = (dog *) malloc (sizeof(dog));
    strcpy(toto->name, "toto");
    toto->year = 3;
    toto->weight = 25;
    printf("Name | Year | Weight\n");
    printf("%s | %d | %f \n", toto->name, toto->year, toto->weight);

    free(toto);

    return 0;
}
```