lec9-struct-linkedlist

April 13, 2018

1 Lecture 9: Structure, Linked list

Zewei Chu 4/13/2018

Some logical operators

- &&
- ||
- !

What is a NULL pointer?

A null pointer has a value reserved for indicating that the pointer does not refer to a valid object.

1.1 Struct

A structure is a collecton of one or more variables, possibly of different types, grouped together under a single name for convenient handling.

Now lets define a structure for a student. We want the following information for a student: - char name[] - unsigned int studentID - double scores[]

In [7]:
```
#include<stdio.h>
#include<string.h>
struct student_info{
    char name[50];
    unsigned int studentID;
    double scores[13];
};

int main(){
    struct student_info a, b, c;
    struct student_info arr[10];
    strcpy(a.name, "aaaaaa");
    a.studentID = 1;
    a.scores[0] = 90;
    a.scores[1] = 80;
    arr[0].studentID = 12;
    arr[1] = a;
    printf("%p %p\n", &(arr[1].name), &(a.name));
```

```
        printf("%s %s\n", arr[1].name, a.name);
        return 0;
    }
```

```
0x7ffee7021210 0x7ffee70218f8
aaaaaa aaaaaa
```

```
In [2]: #include<stdio.h>
        struct point{
            int x;
            int y;
        };

        struct point makepoint(int x, int y){
            struct point temp;
            temp.x = x;
            temp.y = y;
            return temp;
        }

        int main(){
            struct point p = makepoint(1,2);
            printf("%d %d\n", p.x, p.y);
            struct point p2 = p;
            printf("%d %d\n", p2.x, p2.y);

        }
```

```
1 2
1 2
```

1.1.1 typedef

typedef is a way to "rename" a type

```
In [ ]: typedef unsigned int uint;
```

```
In [6]: #include<stdio.h>
        #include<string.h>
        typedef struct{
            char name[50];
            unsigned int studentID;
            double scores[13];
        } student_info;
        int main(){
            student_info s1, s2;
            strcpy(s1.name, "apple");
```

```
    s1.studentID = 1;
    s1.scores[0] = 90;
    s1.scores[1] = 80;
    s2 = s1;
    printf("%s\n", s2.name);
    return 0;
}
```

```
apple
```

1.1.2 Where do I put my struct type declaration?

If multiple files use the struct type, declare the type in a .h file.

1.1.3 Functions with structs

- You can define functions to access (read/write) variables of structs

```
In [2]: #include<stdio.h>
        #include<string.h>
        typedef unsigned int uint;
        typedef struct{
            char name[50];
            unsigned int studentID;
            double scores[13];
        } student_info;
        student_info poplate_student_info(char name[], uint sID){
            student_info s;
            strcpy(s.name, name);
            s.studentID = sID;
            return s;
        }
        int main(){
            student_info s = poplate_student_info("Adam", 1);
            return 0;
        }
```

1.1.4 Nested structures

```
In [10]: #include<stdio.h>
         typedef unsigned int uint;
         typedef struct {
             char first[20];
             char middle;
             char last[20];
             uint studentID;
         } student_info;
         typedef struct {
```

```
    student_info student;
    double scores[30];
} student_class_info;

int main(){
    student_class_info s;
    s.student.middle = 'a';
    s.student.studentID = 1;

}
```

1.2 Linked List

A linked list is a linear data structure where each element is a separate object. Each element (we will call it a node) of a list is comprising of two items - the data and a reference to the next node. The last node has a reference to null. The entry point into a linked list is called the head of the list.

In [13]:
```
#include<stdio.h>
typedef struct _intlist intlist;
struct _intlist{
    int value;
    intlist *next;
};

void print_intlist(intlist* p){
    while (p != NULL){
        printf("%d ", p->value);
        p = p->next;
    }
}
int main(){
    intlist *p1, *p2, *p3, *p4;
    intlist l1, l2, l3, l4;
    p1 = &l1;
    p2 = &l2;
    p3 = &l3;
    p4 = &l4;

    l1.value = 1;
    l2.value = 2;
    l3.value = 3;
    l4.value = 4;

    p1->next = p2;
    p2->next = p3;
    p3->next = p4;
    p4->next = NULL;
```

```
        print_intlist(p1);
        return 0;
    }
```

```
1 2 3 4
```