Copyright Warning

ViSLAB
High-Performance Visualization, Communication, Computing

Object Oriented Design

Week 1-3

Before "Class"

- To use C++ effectively for OO Soft. Dev., you need to understand "class".
- "class" is almost "struct"+alpha.

“class” = “struct” + α ?

- according to C++ spec: the followings are “class”
  - class
  - struct
  - union

What is struct

- struct (structure) is a “named” blob of data

```
struct Monster {

    char name[64];     // the name of the monster
    int power;         // the current power
    int status;        // the current status such as live/dead/paused
};
```

- just like C, C++ uses “block” (compound statement)
  - { ...}

What is struct

- If you know C....

```
struct {
    char letter;
    int key;
    char idx;
    float value;
} myStruct
```

- how all variables be allocated in the memory?

Members of "struct"

- data declared in the struct are "members"

```
struct Monster {

    char name[64];     // the name of the monster
    int power;         // the current power
    int status;        // the current status such as live/dead/paused
};
```

- struct can have other structs as its members

```
struct MyMonsters {
    struct Monster mon1;   //1st monster
    struct Monster mon2;   //2st monster
    struct Monster mon3;   //3t monster
    struct Monster mon4;   //4st monster
}
```

What is "struct" for?

- instructions ➛ functions/methods
- data ➛ variables
- program ➛ {data, instructions}
- real-world ➛ related data are grouped.
  - want to process them in a group.

struct == type

- struct is a type...!= instance.
- after declaring a struct, you need to instantiate.

```
#include <string.h>
#include <iostream.h>

struct Monster {
    char name[64];    // the name of the monster
    int power;        // the current power
    int status;       // the current status such as live/dead/paused
};
```

- The declaration will not allocate a memory space to store an instance of the struct.

Accessing struct members

- struct is a blob of data, but

  you would need to access individual member.

- to access a member of a struct:

```
a_struct_variable.member_name
```

- For instance, to access the name of a monster:

```
a_monster.name
```

Accessing struct members

- A member can be treated as an ordinary variable.
- You can:
  - substitute (=)
  - obtain its address (&)
  - arithmetic operation (+, -) ...if allowed

Accessing struct members

- You can also access each member of a struct, which is also a member of another struct.

```
#include <string.h>
#include <iostream.h>

struct Monster {
    char name[64];    // the name of the monster
    int power;        // the current power
    int status;       // the current status such as live/dead/paused
};

struct MyMonsters {
    struct Monster mon1;   //1st monster
    struct Monster mon2;   //2st monster
    struct Monster mon3;   //3t monster
};
```

Accessing struct members

```
int main(void) {
struct MyMonsters myMon;

strcpy(myMon.mon1.name, "Pikachu");
myMon.mon1.power = 100;
myMon.mon1.status = 0;

strcpy(myMon.mon2.name, "Neko");
myMon.mon2.power = 50;
myMon.mon2.status = 1;

strcpy(myMon.mon3.name, "Kame");
myMon.mon3.power = 30;
myMon.mon3.status = 2;

cout << myMon.mon1.name << "\n";
cout << myMon.mon1.power << "\n";
cout << myMon.mon1.status << "\n";

cout << myMon.mon2.name << "\n";
cout << myMon.mon2.power << "\n";
cout << myMon.mon2.status << "\n";

cout << myMon.mon3.name << "\n";
cout << myMon.mon3.power << "\n";
cout << myMon.mon3.status << "\n";

return 0;
}
```

Substitution

- You can also treat a struct as one datum when you copy it. (shorter and simpler code).

```
#include <string.h>
#include <iostream.h>

struct Monster {
    char name[64];    // the name of the monster
    int power;        // the current power
    int status;       // the current status such as live/dead/paused
};

int main(void) {
    struct Monster myMon, otherMon;

    strcpy(myMon.mon1.name, "Pikachu");
    myMon.mon1.power = 100;
    myMon.mon1.status = 0;

    otherMon = myMon;

    cout << otherMon.mon1.name << "\n";
    cout << otherMon.mon1.power << "\n";
    cout << otherMon.mon1.status << "\n";

    return 0;
}
```

Array of struct

- You can use an array to group primitive data (`int`, `float`)
  - `int[10]`, `float[32]`
- When you declare an array of struct, the specified number of structs are allocated on/in the memory.
- To access a member of each struct:
  - `mons[0].name`

Array of struct

```
    #include <string.h>
#include <iostream.h>

struct Monster {
    char name[64];    // the name of the monster
    int power;        // the current power
    int status;       // the current status such as live/dead/paused
};

int main(void) {
    struct Monster myMons[10];

    strcpy(myMons[0].name, "Pikachu");
    myMons[0].power = 100;
    myMons[0].status = 0;

    strcpy(myMons[1].name, "Neko");
    myMons[1].power = 50;
    myMons[1].status = 1;

    strcpy(myMons[2].name, "Kame");
    myMons[2].power = 30;
    myMons[2].status = 2;

    int i;
    for (i = 0; i < 3; i++) {
        cout << myMons[i].name << "\n";
        cout << myMons[i].power << "\n";
        cout << myMons[i].status << "\n";
    }
    return 0;
}
```

Struct array as a member

```
    #include <string.h>
#include <iostream.h>

struct Monster {
    char name[64];    // the name of the monster
    int power;        // the current power
    int status;       // the current status such as live/dead/paused
};

struct MyMonsters{
    struct Monster monsters[3];
}

int main(void) {
    struct MyMonsters myMons;

    strcpy(myMons.monsters[0].name, "Pikachu");
    myMons.monsters[0].power = 100;
    myMons.monsters[0].status = 0;

    strcpy(myMons.monsters[1].name, "Neko");
    myMons.monsters[1].power = 50;
    myMons.monsters[1].status = 1;

    strcpy(myMons.monsters[2].name, "Kame");
    myMons.monsters[2].power = 30;
    myMons.monsters[2].status = 2;

    int i;
    for (i = 0; i < 3; i++) {
        cout << myMons.monsters[i].name << "\n";
        cout << myMons.monsters[i].power << "\n";
        cout << myMons.monsters[i].status << "\n";
    }
    return 0;
}
```

Pointer for struct

- Just like any other data type, you can use a struct with a pointer.
- use * and &:

```
struct Monster aMon;

strcpy(aMon.name, "Pikachu");
aMon.power = 100;
aMon.status = 0;
.
.
.
struct Monster *pMon;
pMon = &aMon;
.
.
```

Arrow operator

- When you have a pointer of a variable, you can access the actual value of the variable using '*'

```
cout << (*pMon).name << "\n";
cout << (*pMon).power << "\n";
cout << (*pMon).status<< "\n";
```

- You can also use '->' (arrow operator) to access the member of the struct using the pointer.

```
cout << pMon->name << "\n";
cout << pMon->power << "\n";
cout << pMon->status<< "\n";
```

Struct pointer as an argument

- Why would you use a pointer to pass a struct?
  - memory efficient.
  - you only need to pass XXX bits, which is the size of an address.

Struct pointer as an argument

```
    #include <string.h>
#include <iostream.h>

struct Monster {
    char name[64];    // the name of the monster
    int power;        // the current power
    int status;       // the current status such as live/dead/paused
};

void DisplayMonster(struct Monster *pMon);

int main(void) {
    struct Monster aMon;

    strcpy(aMon.name, "Pikachu");
    aMon.power = 100;
    aMon.status = 0;

    DisplayMonster(&aMon);

    return 0;
}

void DisplayMonster(struct Monster *pMon) {
    cout << pMon->name << "\n";
    cout << pMon->power << "\n";
    cout << pMon->status << "\n";

    return;
}
```

Struct as a return value

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

struct Monster {
    char name[64];
    int power;
    int status;
};

struct Monster GetMonster();

int main (int argc, const char * argv[]) {
    struct Monster aMon = GetMonster();

    cout << "\n";
    cout << aMon.name << "\n";
    cout << aMon.power << "\n";
    cout << aMon.status << "\n";

    return 0;
}

struct Monster GetMonster() {
    static struct Monster aMon;

    strcpy(aMon.name, "Pikachu");
    aMon.power = 100;
    aMon.status = 0;

    return aMon;
}
```

Struct pointer as a return value

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

struct Monster {
    char name[64];
    int power;
    int status;
};

struct Monster* GetMonster();

int main (int argc, const char * argv[]) {
    struct Monster *pMon;

    pMon = GetMonster();

    cout << "\n";
    cout << pMon->name << "\n";
    cout << pMon->power << "\n";
    cout << pMon->status << "\n";

    return 0;
}

struct Monster* GetMonster() {
    static struct Monster aMon;

    strcpy(aMon.name, "Pikachu");
    aMon.power = 100;
    aMon.status = 0;

    return &aMon;
}
```

last slide

- We've looked "struct" (familiar in C).
- next week, we will look at "class" based on "struct".