CS162: Introduction to Computer Science II

Week 2: Dynamically Allocated Structure

01/2024

What is in CS162 today?

- Review
- Dynamically allocating structures
- Accessing data of a pointer to a structure

CS162 - Review of Pointers

- What is a pointer?
- How would you define a pointer variable, that can point to a float?
- Would this change if you wanted the pointer to reference an array of floats?
- Show how to dynamically allocate an array of 20 floats
- Show two ways of accessing element 19

CS162 - Review of Pointers

- What operator allocates memory dynamically?
- What does it really mean to allocate memory? Does it have a name?
- Why is it important to subsequently deallocate that memory?
- What operator deallocates memory?

CS162 - Dynamic Structures

- □ Let's take these notions and apply them to dynamically allocated structures
- □ What if we had a video structure, how could the client allocate a video dynamically?

```
video* ptr = new video;
```

- □ Then, how would we access the title?

`*ptr.title`   ? Nope!   WRONG

CS162 - Dynamic Structures

- To access a member of a struct, we need to realize that there is a “precedence” problem.
- Both the dereference (*) and the member access operator (.) have the same operator precedence....and they associate from right to left
- So, parens are required:

`(*ptr).title` Correct (but ugly)

CS162 - Dynamic Structures

□ A short cut (luckily) cleans this up:

`(*ptr).title` Correct (but ugly)

Can be replaced by using the indirect member access operator (->) ... it is the dash followed by the greater than sign:

`ptr->title` Great!

CS162 - Dynamic Structures

- Now, to allocate an array of structures dynamically:

```
video* ptr;
ptr = new video[some_size];
```

- In this case, how would we access the first video's title?

```
ptr[0].title
```

Notice that the -> operator would be incorrect in this case because ptr[0] is not a pointer variable. Instead, it is simply a video object. ptr is a pointer to the first element of an array of video objects

CS162 - Dynamic Structures

- What this tells us is that the -> operator expects a pointer variable as the first operand.
  - In this case, ptr[0] is not a pointer, but rather an instance of a video structure. Just one of the elements of the array!
  - the . operator expects an object as the first operand...which is why it is used in this case!

CS162 - Dynamic Structures

- Ok, what about passing pointers to functions?
- Pass by value and pass by reference apply.
  - Passing a pointer by value makes a copy of the pointer variable (i.e., a copy of the address).
  - Passing a pointer by reference places an address of the pointer variable on the program stack.

CS162 - Dynamic Structures

- Passing a pointer by value:

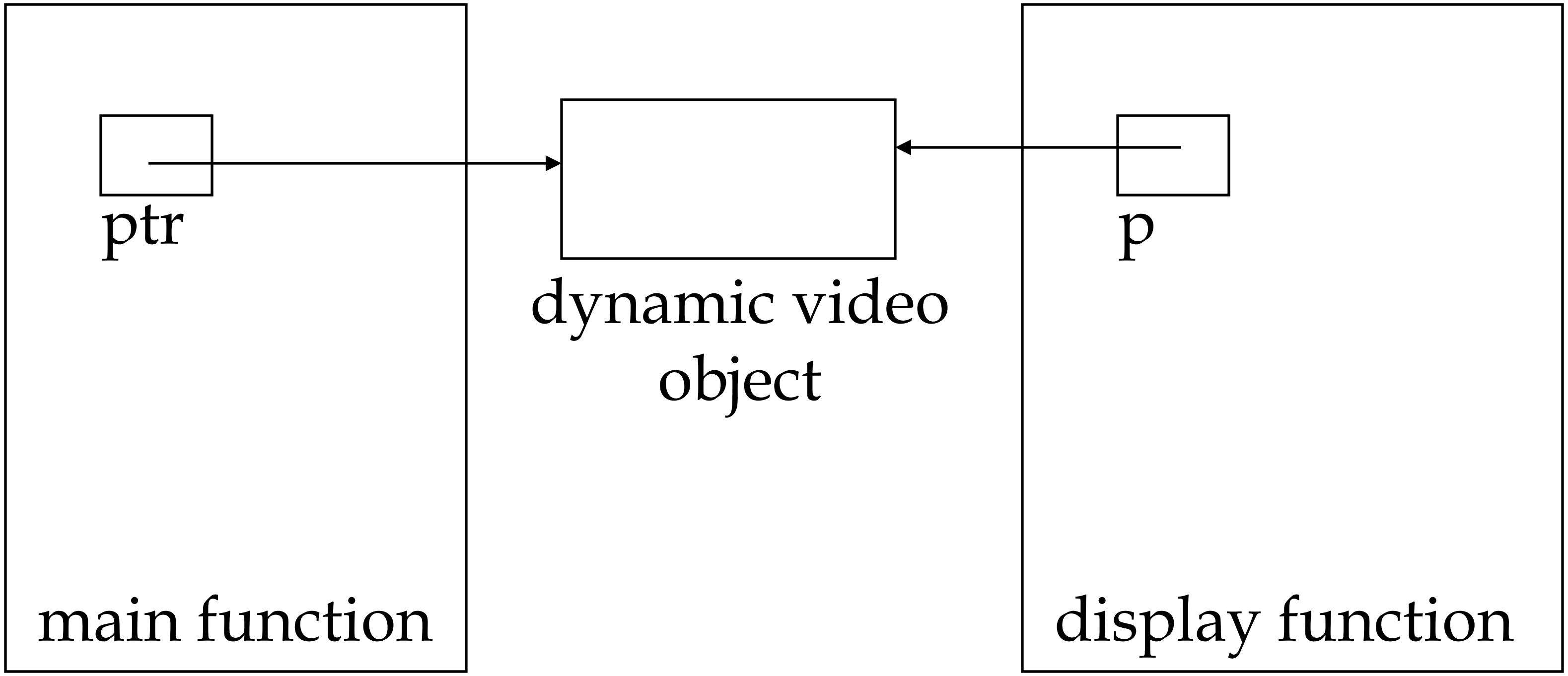
```
    video* ptr = new video;
    display(ptr);
```

```
    void display(video* p)  {
      cout << p->title <<endl;
    }
```

p is a pointer to a video object, passed by value. So, p is a local variable with an initial value of the address of a video object

CS162 - Dynamic Structures

- Here is the pointer diagram for the previous example:



CS162 - Dynamic Structures

- Passing a pointer by reference allows us to modify the calling routine’s pointer variable (not just the memory it references):

```
video* ptr; set(ptr);  cout << ptr->title;

void set(video* &p) {
   p = new video;
   cin.ignore(100,’\n’);
   cin.get(p->title,100,’\n’);
}
```

The order of the * and & is critical!

CS162 - Dynamic Structures

- But, what if we didn’t want to waste memory for the title (100 characters may be way too big (Big, with Tom Hanks)
- So, let’s change our video structure to include a dynamically allocated array:

```
struct video {
  char* title;
  char category[5];
  int quantity;
};
```

CS162 - Dynamic Structures

- Rewriting the set function to take advantage of this:

```
video* ptr;      set(ptr);
```

```
void set(video* &p) {
   char temp[100];
   cin.ignore(100,’\n’);
   cin.get(temp,100,’\n’);
   p = new video;
   p->title = new char[strlen(temp)+1];
   strcpy(p->title,temp);
}
```

watch out for where the +1 is placed!

CS162 - Dynamic Structures

- □ But, what about that list of videos discussed earlier this term?
- □ Let’s write a class that now allocates this list of videos dynamically, at run time
- □ This way, we can wait until we run our program to find out how much memory should be allocated for our video array