CS 162
Intro to CS II

Review: Pointers, Arrays, Structs & Classes

Oregon State University

Array Review…

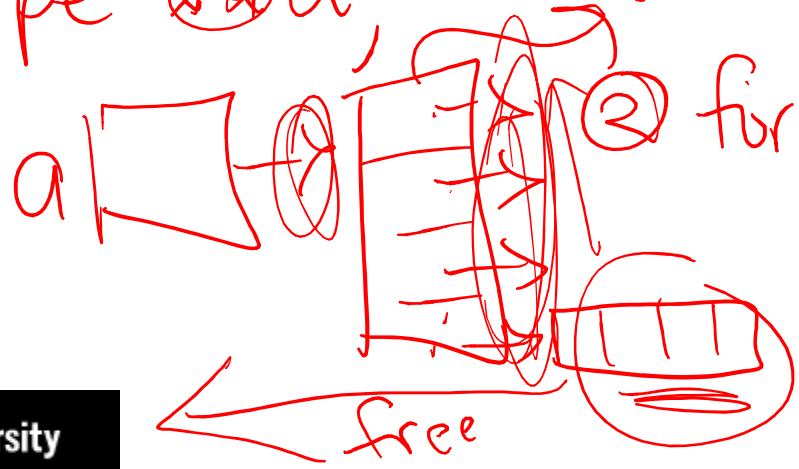
- How do we create a 1-d dynamic array of n?

type *a; a = new type[n];

- How do we create a 2-d dynamic array of m x n?

type **a; ① a = new type *[m];

② for (int i=0; i<m; i++)
a[i] = new type[n];



Array Review…

- How do we free a 1-d?

```
delete [] a;
```

- How do we free a 2-d?

```
for (int i=0; i<m; i++)
    delete [] a[i];
delete [] a;
```

Dynamic Arrays

Array Review…

pointer

- Example Function call: fun(array);
  - Can we change the contents of an array? yes
  - Can we change we change where it points?

C++ depends? we need to know params, if pass by ref

C - no

- Example Function call: fun(&array);
  - Can we change the contents of an array? yes
  - Can we change we change where it points?

yes

access.engr.orst.edu - PuTTY

```
 1 #include <iostream>
 2 #define M 4
 3 #define N 6
 4 using namespace std;
 5
 6 void del(int ***a){
 7     for(int i=0; i<M; i++)
 8         delete [] (*a)[i];
 9     delete [] (*a);
10
11     *a=NULL; //Set back to NULL after delete
12 }
13 int main() {
14     int **a=NULL;
15
16     a = new int*[M];
17     for(int i=0; i<M; i++)
18         a[i]=new int[N];
19
20     del(&a); //Pass address of a to match int ***
21
22     return 0;
23 }
-- INSERT --                                  21,1          Top
```

Struct Review…

- What is a struct? container, bundle, record.
- How do we create the type? globally – using struct Keyword
- How are they passed by default?
  not pointers.

- What if we want to change contents?

fun(s);

by ref in params

type fun (struct s_type &s);

s.

fun(&s);

type fun(struct s_type *)

s → deref

(*s).

Structs vs. Classes

- Differences

functionality in classes!

* structs - members are public
* classes - members are private

- Similarities

contain several diff. types of data.

Class vs. Object

hiThere Java
hi_there C/C++

- Class declaration is type.
- Object is an instance of a class.
- Example:

```
class Point {
public:
  int x;
  int y;
};
int main() {
    Point p1, p2;

    p1.x=10;   p1.y=20;
    p2.x=5;   p2.y=6;

    return 0;
}
```

class declaration

objects setup in memory

dot operator

Class w/ Behavior/Member Functions

```
class Point {
public:
  int x;
  int y;
  void translate(int dx, int dy);  //Translates to a new x, y location given distance
};
int main () {
  Point p1, p2;
  p1.x=10;   p1.y=20;
  p2.x=5;   p2.y=6;

  p1.translate(-1, 3);
  p2.translate(2, -2);
  return 0;
}
void Point::translate(int dx, int dy) {
     x += dx;
     y += dy;
}
```

Scope operator

Can we set the values for x and y?

```
class Point {
public:
  int x = 0;    //This is not allowed!!!
  int y = 0;    //This is not allowed!!!
  void translate(int dx, int dy);  //Translates to a new x, y location given distance
};
int main () {
  Point p1, p2;
  p1.x=10;   p1.y=20;
  p2.x=5;   p2.y=6;

  p1.translate(-1, 3);
  p2.translate(2, -2);
  return 0;
}
void Point::translate(int dx, int dy) {
     x += dx;
     y += dy;
}
```

What if we made states private?

```
class Point {
public:
   void translate(int dx, int dy);
private:
   int x;
   int y;
};
int main () {
   Point p1, p2;
   p1.x=10;   p1.y=20;    //This is not allowed!!!
   p2.x=5;   p2.y=6;       //This is not allowed!!!

   p1.translate(-1, 3);
   p2.translate(2, -2);
   return 0;
}
void Point::translate(int dx, int dy) {
      x += dx;
      y += dy;
}
```

cannot directly access

Encapsulation/ADTs

abstract data type

- Why do this?
- How do we set/get private member variables?
  - Accessor and Mutator Functions
- Access: get…
- Mutator: set…

Encapsulation

changing member value

```
class Point {
public:
  void set_xy(int theX, int theY);   //Mutator Function
private:
  int x;
  int y;
};
int main () {
  Point p1, p2;
  p1.set_xy(1, 1);
  p2. set_xy(2, 2);
  return 0;
}
void Point::set_xy(int theX, int theY) {
      x = theX;
      y = theY;
}
```

How do we write an Accessor Function?

```
class Point {
public:
  void set_xy(int theX, int theY);   //Mutator Function
  int get_x();   //Accessor Function
  int get_y();   //Accessor Function
private:
  int x;
  int y;
};
int main () {
  Point p1;
  p1.set_xy(1, 1);
  std::cout << p1.get_x(); << "\t" << p1.get_y() << "\n";
  return 0;
}
int Point::get_x() {
     return x;   }
int Point::get_y() {
     return y;   }
void Point::set_xy(int theX, int theY) {
    ...}
```

return member values.

Always define

Constructors…

① name of function is same as class name

② no return type

```
class Point {
public:
  Point(int x_val, int y_val);  //Constructor
  void set_xy(int theX, int theY);  //Mutator Function
  int get_y();  //Accessor Function
  int get_x();  //Accessor Function
private:
  int x;
  int y;
};
int main () {
  Point p1(1, 1), p2(2, 2);
  p1.Point(1, 1);  //This is illegal
  p2.Point(2, 2);  //This is illegal
  return 0;
}
Point::Point(int x_val, int y_val) {
    x=x_val;  y=y_val;
}
```

Constructors…

```
class Point {
public:
  Point(int x_val, int y_val);   //Constructor
  void set_xy(int theX, int theY);   //Mutator Function
  int get_y();   //Accessor Function
  int get_x();   //Accessor Function
private:
  int x;
  int y;
};
int main () {
  Point p1, p2;   //Calls the default constructor but we don't have one!!!
  return 0;
}
Point::Point(int x_val, int y_val) {
    x=x_val;   y=y_val;
}
```

Constructors…

```
class Point {
public:
  Point();
  Point(int x_val, int y_val);   //Constructor
  void set_xy(int theX, int theY);   //Mutator Function
  int get_y();   //Accessor Function
  int get_x();   //Accessor Function
private:
  int x;
  int y;
};
int main () {
  Point p1, p2(2,4);
  return 0;
}
Point::Point(int x_val, int y_val) {
    x=x_val;   y=y_val;
}
Point::Point() { x=0; y=0; }
```

default constructor

Another way to define Constructors…

```
class Point {
public:
  Point(int x_val, int y_val);   //Constructor
  void set_xy(int theX, int theY);   //Mutator Function
  int get_y();   //Accessor Function
  int get_x();   //Accessor Function
private:
  int x;
  int y;
};
int main () {
  Point p1(1, 1), p2(2, 2);
  return 0;
}
Point::Point(int x_val, int y_val)
      :x(x_val), y(y_val)
{ /*Do nothing in here*/ }
```

More defining Constructors…

```
class Point {
public:
  Point(int x_val, int y_val);   //Constructor
  void set_xy(int theX, int theY);   //Mutator Function
  int get_y();   //Accessor Function
  int get_x();   //Accessor Function
private:
  int x;
  int y;
};
int main () {
  Point p1(1, 1), p2(2, 2);
  return 0;
}
Point::Point(int x_val, int y_val)
        : x(x_val), y(y_val)
{
  if(x < 0 || y < 0)
    std::cout << "You need to enter a positive number" << std::endl;
}
```

do something other than just initialize