CS 31 Week 10
Discussion 2E

Srinath

Announcements

- **The final is on Saturday, December 3, from 11:30 to 2:30**
- **Fill out the course and TA evaluation - Your feedback helps!**

Outline

- Review

Topics

- Strings
- If statement
- Loops
- Functions
- Arrays
- C Strings
- Pointers
- Structs/Classes

Strings

How to read in a string from user input?

Strings

How to read in a string from user input?

- Use **getline(cin, s);**
- Let's say the user typed "Hello world" and then Enter(i,e newline)
- All of the string along with newline will be consumed by the program i,e "Hello world\n"
- But, **s** doesn't contain newline. i,e **s** will be "Hello world"

Strings

cin.ignore(10, ‘\n’);
What does this do?

Strings

cin.ignore(10, ‘\n’);

What does this do?

- Ignores characters available to program until ‘\n’(including it) is reached or a maximum of 10 characters.
- ‘abcd\n’ - all of it will be ignored
- ‘abcdefghijkl\n’ - only ‘abcdefghij’ will be ignored.

When should we use it?

Strings

cin.ignore(10, '\n');

What does this do?

- Ignores characters available to program until '\n'(including it) is reached or a maximum of 10 characters.
- 'abcd\n' - all of it will be ignored
- 'abcdefghijkl\n' - only 'abcdefghij' will be ignored.

When should we use it?

- *only* time to use is after reading a number and the next to read is a string.

```
cout << "How many courses you enrolled this term?"<<endl;
int numCoursesEnrolled;
cin >> numCoursesEnrolled;
cin.ignore(10000, '\n');

cout << "What is your favourite course among them?"<<endl;
string favCourse;
getline(cin, favCourse);
```

What will happen if we don't use it?

Strings

cin.ignore(10, '\n');

What does this do?

- Ignores characters available to program until '\n'(including it) is reached or a maximum of 10 characters.
- 'abcd\n' - all of it will be ignored
- 'abcdefghijkl\n' - only 'abcdefghij' will be ignored.

When should we use it?

- *only* time to use is after reading a number and the next to read is a string.

```
cout << "How many courses you enrolled this term?"<<endl;
int numCoursesEnrolled;
cin >> numCoursesEnrolled;
cin.ignore(10000, '\n');

cout << "What is your favourite course among them?"<<endl;
string favCourse;
getline(cin, favCourse);
```

What will happen if we don't use it?

- Empty string will be read into the variable favCourse because '\n' entered for reading the int is still available to the program.

If statement

If statement

- If …
- If … else ….
- If … else If …. else If … (If - else ladder)

Switch statement

- switch(var) { case A:, case B: .. etc.. }
- Recall about use of **break;**
- **default** case
- Combining multiple cases

Switch :

```
switch(expression) {
   case constant-expression  :
      statement(s);
      break; //optional
   case constant-expression  :
      statement(s);
      break; //optional

   // you can have any number of case statements.
   default : //Optional
      statement(s);
}
```

default is executed if expression do not match any of the cases.

Switch :

```
switch(expression) {
   case constant-expression  :
      statement(s);
      break; //optional
   case constant-expression  :
      statement(s);
      break; //optional

   // you can have any number of case statements.
   default : //Optional
      statement(s);
}
```

default is executed if expression do not match any of the cases.

```
int vehicleClass;
double toll;
cout << "Enter vehicle class: ";
cin >> vehicleClass;
switch (vehicleClass)
{
   case 1:
      cout << "Passenger car.\n";
      toll = 0.50;
      break;
   case 2:
      cout << "Bus.\n";
      toll = 1.50;
      break;
   case 3:
      cout << "Truck.\n";
      toll = 2.00;
      break;
   default:
      cout << “Unknown Vehicle.\n”;
   }
```

What if we remove break statement in case 2 and the input vehicleClass is 2?

Switch :

```
switch(expression) {
   case constant-expression  :
      statement(s);
      break; //optional
   case constant-expression  :
      statement(s);
      break; //optional

   // you can have any number of case statements.
   default : //Optional
      statement(s);
}
```

default is executed if expression do not match any of the cases.

```
int vehicleClass;
double toll;
cout << "Enter vehicle class: ";
cin >> vehicleClass;
switch (vehicleClass)
{
   case 1:
      cout << "Passenger car.\n";
      toll = 0.50;
      break;
   case 2:
      cout << "Bus.\n";
      toll = 1.50;
      break;
   case 3:
      cout << "Truck.\n";
      toll = 2.00;
      break;
   default:
      cout << “Unknown Vehicle.\n”;
   }
```

What if we remove break statement in case 2 and the input vehicleClass is 2?

- Bus
  Truck

- **toll** will be 2.00

Loops

Loops

- While
- **Do … While**
- For …

```
while(condition) {
   statement(s);
}
```

```
do {
   statement(s);
}
while( condition );
```

```
for ( init; condition; increment ) {
   statement(s);
}
```

Looping a String :

As a string is a sequence of characters, we can loop through the string and access each of its characters in an orderly fashion.

```
string name = “course-cs31”;
for(int i=0 ; i != name.size(); i++) {
        cout << name.at(i) <<endl;
}
```

Prints the string, character by character

Counting number of A’s (both lower and upper case)

```
string name = “cs31 is awesome!”;
int numOfAs = 0;
for(int i=0 ; i != name.size(); i++) {
        If (name.at(i)==‘a’ || name.at(i)==‘A’ ) {
                numOfAs = numOfAs+1;
        }
}

cout << numOfAs << endl;
```

Functions

Functions

- Declaring a function(Just the declaration)
- Defining a function(implementation)
- Return types, argument types, return values
- Pass by value
- Pass by reference

Functions : Examples

```
int cube(int x) {
        int result = x*(x*x);
}
```

Will this compile?

```
string cube(int x) {
        int result = x*(x*x);
        return result;
}
```

Will this compile?

```
int cube(int x) {
        int result = x*(x*x);
        return -1;
}
```

Will this compile?
Does it do what we need?

```
int cube(int x) {
        int result = x*(x*x);
        return result;
}
```

Functions : Examples

```
int cube(int x) {
        int result = x*(x*x);
}
```

Will this compile? - No

```
string cube(int x) {
        int result = x*(x*x);
        return result;
}
```

Will this compile? - No

```
int cube(int x) {
        int result = x*(x*x);
        return -1;
}
```

Will this compile? - Yes
Does it do what we need? - No

```
int cube(int x) {
        int result = x*(x*x);
        return result;
}
```

Compiles and does what we need.

Functions : Pre-defined

C++ has libraries with defined functions for you.

Some other examples?

isdigit()
isalpha()
isupper()
islower()
Above are included with header <cctype>

size()
substr(..)
etc..

All these predefined functions require `using namespace std;` as well as an `include` directive.

NAME	DESCRIPTION	TYPE OF ARGUMENTS	TYPE OF VALUE RETURNED	EXAMPLE	VALUE	LIBRARY HEADER
sqrt	Square root	double	double	sqrt(4.0)	2.0	cmath
pow	Powers	double	double	pow(2.0,3.0)	8.0	cmath
abs	Absolute value for int	int	int	abs(-7) abs(7)	7 7	cstdlib
labs	Absolute value for long	long	long	labs(-70000) labs(70000)	70000 70000	cstdlib
fabs	Absolute value for double	double	double	fabs(-7.5) fabs(7.5)	7.5 7.5	cmath
ceil	Ceiling (round up)	double	double	ceil(3.2) ceil(3.9)	4.0 4.0	cmath
floor	Floor (round down)	double	double	floor(3.2) floor(3.9)	3.0 3.0	cmath
exit	End program	int	void	exit(1);	None	cstdlib
rand	Random number	None	int	rand( )	Varies	cstdlib
srand	Set seed for rand	unsigned int	void	srand(42);	None	cstdlib

Functions :

```
int main() {
        cout<< “square of 2 is” << square(2) <<endl;
        cout<< “square of 20 is” << square(20) <<endl;
        cout<< “square of 200 is” << square(200) <<endl;
        int n = 23456;
        int n_squared = square(n);
        return 0;
}

int square(int x) {
        int result = x*x;
        return result;
}
```

Will this compile?

Functions :

```
int main() {
        cout<< “square of 2 is” << square(2) <<endl;
        cout<< “square of 20 is” << square(20) <<endl;
        cout<< “square of 200 is” << square(200) <<endl;
        int n = 23456;
        int n_squared = square(n);
        return 0;
}

int square(int x) {
        int result = x*x;
        return result;
}
```

Will this compile?

- No, it doesn’t know what square(..) is

Functions : Declaration

```
int square(int x); // declaration

int main() {
        cout<< “square of 2 is” << square(2) <<endl;
        cout<< “square of 20 is” << square(20) <<endl;
        cout<< “square of 200 is” << square(200) <<endl;
        int n = 23456;
        int n_squared = square(n);
        return 0;
}

int square(int x) {
        int result = x*x;
        return result;
}
```

Will this compile?

- No, it doesn’t know what square(..) is

For this to compile, you have to **declare** the function before using it.

C++ provides a way to declare function and later write its implementation.

returnType functionName(arg1Type arg1Name, arg2Type arg2Name, …) ;

Functions : Practice

```
#include <iostream>
using namespace
std;

// function declaration
void swap(int x, int
y);

int main () {
   // local variable declaration:
   int a =
   100;  int b
   = 200;

   cout << "Before swap, val of a :" << a <<
   endl;<<cout << "After swap, val of a :" << b
   <<endl;cout << "Before swap, val of b :"
   << endl;cout << "After swap, val of b :" << b
   << endl;

   return 0;
}
```

```
// swap the values of two numbers
void swap(int x, int y) {
    int temp;
    temp = x;  /* save the value of x */
    x = y;  /* put y into x */
    y = temp;  /* put x into y */
    return;
}
```

Output ??

Functions : Practice

```
#include <iostream>
using namespace
std;

// function declaration
void swap(int x, int
y);

int main () {
   // local variable declaration:
   int a =
   100;  int b
   = 200;

   cout << "Before swap, val of a :" << a <<
   endl;<<cout << "Before swap, val of b :" << b
   <<endl;cout << "After swap, val of a :" <<
   endl;cout << "After swap, val of b :" << b
   << endl;

   return 0;
}
```

```
// swap the values of two
numbers  void swap(int x, int y)
{
    int temp;
    temp = x; /* save the value of x */
    x = y; /* put y into x */
    y = temp; /* put x into y
    */  return;
}
```

Output ??

Before swap, val of a :100
Before swap, val of b :200
After swap, val of a :100
After swap, val of b :200

Functions : Practice

```
#include <iostream>
using namespace
std;
// function declaration
void swap(int &x, int &y);

int main () {
   // local variable
   declaration:  int a = 100;
   int b = 200;
   cout << "Before swap, val of a :" << a <<
   endl;  cout << "Before swap, val of b :" << b
   << endl;

   swap(a,b);
   cout << "After swap, val of a :" << a <<
   endl;  cout << "After swap, val of b :" << b
   << endl;
   return
} 0;
```

```
// swap the values of two
void
variables       &x, int &y) {
   swap(int
   int temp; /* save the value of x
   temp = x; */
   x = y;      /* put y into x */
   y       = /* put x into y */
   temp;
   return;
}
```

Output ??

Functions : Practice

```
#include <iostream>
using namespace
std;
// function declaration
void swap(int &x, int &y);

int main () {
   // local variable
   declaration:  int a = 100;
   int b = 200;
   cout << "Before swap, val of a :" << a <<
   endl;  cout << "Before swap, val of b :" << b
   << endl;

   swap(a,b);
   cout << "After swap, val of a :" << a <<
   endl;  cout << "After swap, val of b :" << b
   << endl;
   return
}  0;
```

```
// swap the values of two
void
variables        &x, int &y) {
   swap(int
   int temp; /* save the value of x
   temp = x; */
   x = y;      /* put y into x */
   y       = /* put x into y */
   temp;
   return;
}
```

Output ??

Before swap, val of a :100
Before swap, val of b :200
After swap, val of a :200
After swap, val of b :100

Arrays

Arrays

- Declaration
- Initialization
- Access
- Looping
- Passing as parameters

Arrays : Defining

Initialize

int duration [5];
If initializer is not specified, value will be some garbage.

int heights [5] = { };
The values will be initialized with default value.

double weights [] = {10, 5, 6, 19, 21}
If you do not specify size, compiler can infer it.

```
#include <iostream>
using namespace std;

void print(int arr[], int n) {
    // prints elements of the array
}

int main() {
    cout << "Hello World" << endl;
    int N=5;
    int arr[N];
    print(arr, N);
    int arr1[5]={9, 8, 7, 6, 5};
    print(arr1, N);
    int arr2[5] = {};
    print(arr2, N);
    int arr3[5] = {13, 14, 15};
    print(arr3, N);
}
```

Output :

```
Hello World
-516134080 32766 243536080 1 -516134208
9 8 7 6 5
0 0 0 0 0
13 14 15 0 0
```

Arrays : as function parameters

```
double average(int heights[ ], int size) {
        //.. do whatever with the array
}
```

Why do we pass size separately?

Calling the above function

```
int classHeights [ ] = {1, 2, 3, 4, 5};
double averageHeight = average(classHeights, 5);
cout << averageHeight << endl;
```

When calling the function with array as parameter
Is the array copied? -
Is it pass by value? -
Will the original array get modified if its modified in the function? -

Arrays : as function parameters

```
double average(int heights[ ], int size) {
        //.. do whatever with the array
}
```

Calling the above function

```
int classHeights [ ] = {1, 2, 3, 4, 5};
double averageHeight = average(classHeights, 5);
cout << averageHeight << endl;
```

Passing arrays to functions in C/C++ are passed by reference.

Why do we pass size separately?
In C++ there is no standard way of getting the array size, so always need a separate variable to keep track of it.

Remember to pass size too.

When calling the function with array as parameter
Is the array copied? - No
Is it pass by value? - No
Will the original array get modified if its modified in the function? - Yes
It's pass by reference - notice that we didn't need the '&' before.

2D Arrays : as Function parameters

```
void processArr(int a[][2], int n_rows, int n_cols) {
        cout << "element at index 1,1 is " << a[1][1];
}

int main() {
   int arr[2][2];
   arr[0][0] = 0;
   arr[0][1] = 1;
   arr[1][0] = 2;
   arr[1][1] = 3;

   processArr(arr, 2, 2);
   return 0;
}
```

leave off the first bound, but you must supply the second bound as a compile-time constant expression

For multidimensional arrays, leave off first, but must supply the rest of bounds.
Eg: **process(int c[][5][7]);**

C Strings

C Strings

- Declaration
- Initialization
- strlen - length
- strcmp - compare
- strcpy - copy
- strcat - concatenate

C Strings

Declaration

```
char name[20];
```

Is this a C String right now?

Initialization

```
char title[10] = {'w', 'o', 'r', 'l', 'd', '\0'};   // standard array initialization

char course[13] = "computer";                       // shortcut array initialization

char role[13] = "";                                 // empty string, length = 0
```

What will be the remaining values?

```
char str1[ ] = {'c', 'h', 'e', 's', 's', '\0'};     // without size specification
char str[ ] = "hello";                              // without size specification
```

Is this valid?

```
char title[10] = {'\0', 'w', 'o', 'r', 'l', 'd'};
```

What is the length here?

C Strings

Declaration

```
char name[20];    // - can hold C-String, but is not yet a valid C-String
```

Is this a C String right now?
- No

Initialization

```
char title[10] = {'w', 'o', 'r', 'l', 'd', '\0'};    // standard array initialization

char course[13] = "computer";                        // shortcut array initialization

char role[13] = "";                                  // empty string, length = 0
```

What will be the remaining values?
- set to value of '\0'

```
char str1[ ] = {'c', 'h', 'e', 's', 's', '\0'};    // without size specification
char str[ ] = "hello";                             // without size specification
```

Is this valid?
- Yes

```
char title[10] = {'\0', 'w', 'o', 'r', 'l', 'd'};
```

What is the length here?
- 0
- equivalent to empty string

Pointers

Pointers

- Declaration, Initialization
- *, & operators
- Pointers with arrays
- Pointers with functions
- Pointer axioms/arithmetic.

Pointers : Axioms

Some axioms(true by default) on pointers and pointer arrays

*&x ⇒ x
&a[i] + j ⇒ &a[i+j]
&a[i] - j ⇒ &a[i-j]
&a[i] < &a[j] ⇒ i < j (also for <=, >, >=, ==, !=)
&a[i] - &a[j] ⇒ i - j
p[i] <==> *(p+i)

Structs/Classes

Struct/Classes

- Defining, Declaration, Initialization, '.' dot operation
- Pointer to struct, '->' operator
- Public/Private, Class vs Struct
- Member functions : inside struct, outside struct, const member functions
- Constructors : Default, with arguments
- Dynamic allocation, 'delete' operator
- Destructor
- Structs with Functions
- Array of Structs, Struct Pointers

Struct : Constructor

Can we define our own constructor?

-

If we write our own constructor, will the compiler still write a default one for us?

-

Can we define constructor with parameters?

-

Can we define multiple constructors?

-

Does ‘Employee’ struct have a default constructor?

-

```
struct Employee {
        string m_name;
        double m_salary;
    private:
        int m_age;

    public:
    Employee(string name){
        m_name = name;
    }

    Employee(int age){
        m_age = age;
    }
};
```

Struct : Constructor

Can we define our own constructor?
- *Yes!*

If we write our own constructor, will the compiler still write a default one for us?
- *No!*

Can we define constructor with parameters?
- *Yes!*

Can we define multiple constructors?
- *Yes!*

Does ‘Employee’ struct have a default constructor?
- *No. Default is one is with no parameters*
  => Employee() {...}

```
struct Employee {
        string m_name;
        double m_salary;
    private:
        int m_age;

    public:
    Employee(string name){
        m_name = name;
    }

    Employee(int age){
        m_age = age;
    }
};
```

Struct : Constructor

```
Employee emp;          // Will this compile?
-
Employee emp("jack");  // Will this compile?
-
Employee emp(100);     // Will this compile?
-
Employee emp(200.0);   // Will this compile?
-
```

```
struct Employee {
        string m_name;
        double m_salary;
    private:
        int m_age;

    public:
    Employee(string name){
        m_name = name;
    }

    Employee(int age){
        m_age = age;
    }
};
```

Struct : Constructor

Employee emp; // Will this compile?
- No, it doesn't have a default constructor.

Employee emp("jack"); // Will this compile?
- Yes.

Employee emp(100); // Will this compile?
- Yes.

Employee emp(200.0); // Will this compile?
- No, it doesn't have constructor with double as a parameter.

```
struct Employee {
        string m_name;
        double m_salary;
    private:
        int m_age;

    public:
    Employee(string name){
        m_name = name;
    }

    Employee(int age){
        m_age = age;
    }
};
```

Struct : Constructor

```
Employee emp;          // Will this compile?
```

```
struct Employee {
        string m_name;
        double m_salary;
    private:
        int m_age;

    public:
    Employee(){
        m_salary = 0.0;
        m_age = 1;
    }

    Employee(string name){
        m_name = name;
    }

    Employee(int age){
        m_age = age;
    }
};
```

Struct : Constructor

```
Employee emp;          // Will this compile?
```

- Yes.

```
struct Employee {
        string m_name;
        double m_salary;
    private:
        int m_age;

    public:
    Employee(){
        m_salary = 0.0;
        m_age = 1;
    }

    Employee(string name){
        m_name = name;
    }

    Employee(int age){
        m_age = age;
    }
};
```

Struct : Dynamic allocation

Dynamic objects are which created during run-time.
These are stored in the heap-memory.

Most of the times we might need to create objects during run-time as we do not know full details during compile time.

How can we create a dynamic struct/class object?

```
struct Employee {
        string m_name;
        double m_salary;
    private:
        int m_age;

    public:
    Employee(){
        m_salary = 0.0;
        m_age = 1;
    }

    Employee(string name){
        m_name = name;
    }

    Employee(int age){
        m_age = age;
    }
};
```

Struct : Dynamic allocation

Dynamic objects are which created during run-time.
These are stored in the heap-memory.

Most of the times we might need to create objects during run-time as we do not know full details during compile time.

How can we create a dynamic struct/class object?

- Using ‘new’ declaration

*Employee * p = new Employee();*

note that new always returns a pointer to the object created.

```
struct Employee {
        string m_name;
        double m_salary;
    private:
        int m_age;

    public:
    Employee(){
        m_salary = 0.0;
        m_age = 1;
    }

    Employee(string name){
        m_name = name;
    }

    Employee(int age){
        m_age = age;
    }
};
```

Struct : Dynamic allocation

```
int main() {
        Employee * p = new Employee("Jill");
        p->m_salary = 256.0;
        return 0;
}
```

Is something not right with our program?

```
struct Employee {
        string m_name;
        double m_salary;
    private:
        int m_age;

    public:
    Employee(){
        m_salary = 0.0;
        m_age = 1;
    }

    Employee(string name){
        m_name = name;
    }

    Employee(int age){
        m_age = age;
    }
};
```

Struct : Dynamic allocation

```
int main() {
        Employee * p = new Employee("Jill");
        p->m_salary = 256.0;
        return 0;
}
```

Is something not right with our program?

- Memory leaks, we haven't deleted the memory allocated.

```
struct Employee {
        string m_name;
        double m_salary;
    private:
        int m_age;

    public:
    Employee(){
        m_salary = 0.0;
        m_age = 1;
    }

    Employee(string name){
        m_name = name;
    }

    Employee(int age){
        m_age = age;
    }
};
```

Struct : Dynamic allocation

```
int main() {
        Employee * p = new Employee("Jill");
        p->m_salary = 256.0;
        delete p;
        return 0;
}
```

Is something not right with our program?

- Memory leaks, we haven't deleted the memory allocated.

Use 'delete' to free up the memory allocated.

- Call 'delete' only on dynamically allocated object's pointer.

```
struct Employee {
        string m_name;
        double m_salary;
    private:
        int m_age;

    public:
    Employee(){
        m_salary = 0.0;
        m_age = 1;
    }

    Employee(string name){
        m_name = name;
    }

    Employee(int age){
        m_age = age;
    }
};
```

Struct : Const Member Functions

```
int main() {
        UC ucla;
        Employee emp("Einstein");
        cout<< ucla.isOnStrike(emp) << endl;
}
```

Will this compile?

-

```
struct Employee {
        string m_name;
    private:
        double m_salary;
        int m_age;

    public:
    Employee(string name){
        m_name = name;
        m_salary = 0.0;
        m_age = 1;
    }

    double getSalary() const {
        m_salary++;
        return m_salary;
    }
};

struct UC {
    double m_budget;
    public:
    UC(){m_budget = 9000.0}

    bool isOnStrike(const Employee * emp){
        double sal = emp->getSalary();
        return sal<100.0;
    }
}
```

Struct : Const Member Functions

```
int main() {
        UC ucla;
        Employee emp("Einstein");
        cout<< ucla.isOnStrike(emp) << endl;
}
```

Will this compile?

- No, const member function shouldn't modify any member variable.

```
struct Employee {
        string m_name;
    private:
        double m_salary;
        int m_age;

    public:
    Employee(string name){
        m_name = name;
        m_salary = 0.0;
        m_age = 1;
    }

    double getSalary() const {
        m_salary++;
        return m_salary;
    }
};

struct UC {
    double m_budget;
    public:
    UC(){m_budget = 9000.0}

    bool isOnStrike(const Employee * emp){
        double sal = emp->getSalary();
        return sal<100.0;
    }
}
```

Struct : with Arrays

You can also declare member variables which are arrays of type struct/class

```
int main(){
        UC ucla;
        ucla.addEmployee("Alice");
        ucla.addEmployee("Bob");
        return 0;
};
```

Is something not right with our program?

-

How can we fix this?

-

```
struct Employee {
        string m_name;
    private:
        double m_salary;
        int m_age;

    public:
    Employee(string name){
        m_name = name;
        m_salary = 0.0;
        m_age = 1;
    }

    double getSalary() const {
        return m_salary;
    }
};

Struct UC {
    Employee * emp_ps[100];
    int n=0;

    addEmployee(string name){
        Employee * p = new Employee(name);
        emp_ps[n] = p;
        n++;
    }

};
```

Struct : with Arrays

You can also declare member variables which are arrays of type struct/class

```
int main(){
        UC ucla;
        ucla.addEmployee("Alice");
        ucla.addEmployee("Bob");
        return 0;
};
```

Is something not right with our program?

- Memory leaks, we haven't deleted the memory allocated.

How can we fix this?

- Destructors

```
struct Employee {
        string m_name;
    private:
        double m_salary;
        int m_age;

    public:
    Employee(string name){
        m_name = name;
        m_salary = 0.0;
        m_age = 1;
    }

    double getSalary() const {
        return m_salary;
    }
};

Struct UC {
    Employee * emp_ps[100];
    int n=0;

    addEmployee(string name){
        Employee * p = new Employee(name);
        emp_ps[n] = p;
        n++;
    }

};
```

Struct : Destructor

The fix to our problem

```
int main(){
        UC ucla;
        ucla.addEmployee("Alice");
        ucla.addEmployee("Bob");
        return 0;
};
```

Destructor of a class is called when the object goes out of scope or when 'delete' is called on its pointer.

```
struct Employee {
        string m_name;
    private:
        double m_salary;
        int m_age;

    public:
    Employee(string name){
        m_name = name;
        m_salary = 0.0;
        m_age = 1;
    }

    ~Employee(){} // default destructor.
};

Struct UC {
    Employee * emp_ps[100];
    int n=0;

    addEmployee(string name){
        Employee * p = new Employee(name);
        emp_ps[n] = p;
        n++;
    }

    ~UC() {
        while(n-1>0){
            delete emp_ps[n-1];
            n- -;
        }
    }
}
```

Struct : Destructor

The fix to our problem

```
int main(){
        UC ucla;
        ucla.addEmployee("Alice");
        ucla.addEmployee("Bob");
        return 0;
};
```

Can we define destructor with parameters?

-

Can we define multiple destructors?

-

Destructor of a class is called when the object goes out of scope or when 'delete' is called on its pointer.

```
struct Employee {
        string m_name;
    private:
        double m_salary;
        int m_age;

    public:
    Employee(string name){
        m_name = name;
        m_salary = 0.0;
        m_age = 1;
    }

    ~Employee(){} // default destructor.
};

Struct UC {
    Employee * emp_ps[100];
    int n=0;

    addEmployee(string name){
        Employee * p = new Employee(name);
        emp_ps[n] = p;
        n++;
    }

    ~UC() {
        while(n-1>0){
            delete emp_ps[n-1];
            n- -;
        }
    }
}
```

Struct : Destructor

The fix to our problem

```
int main(){
        UC ucla;
        ucla.addEmployee("Alice");
        ucla.addEmployee("Bob");
        return 0;
};
```

Can we define destructor with parameters?

- *No!, we don't need them*

Can we define multiple destructors?

- *No! Every class/struct must have one and only one.*

Destructor of a class is called when the object goes out of scope or when 'delete' is called on its pointer.

```
struct Employee {
        string m_name;
    private:
        double m_salary;
        int m_age;

    public:
    Employee(string name){
        m_name = name;
        m_salary = 0.0;
        m_age = 1;
    }

    ~Employee(){} // default destructor.
};

Struct UC {
    Employee * emp_ps[100];
    int n=0;

    addEmployee(string name){
        Employee * p = new Employee(name);
        emp_ps[n] = p;
        n++;
    }

    ~UC() {
        while(n-1>0){
            delete emp_ps[n-1];
            n- -;
        }
    }
}
```

Project 7

```
…. :: moveRabbits(){
        for(int i=0; i<m_nRabbits: i++){
                m_rabbits[i]->move();
                if(m_rabbits[i]->isDead()){
                        delete m_rabbits[i];
                        m_nRabbits - -;
                }
        }
};
```

Project 7

```
…. :: moveRabbits(){
        for(int i=0; i<m_nRabbits: i++){
                m_rabbits[i]->move();
                if(m_rabbits[i]->isDead()){
                        delete m_rabbits[i];

                        m_rabbits[i] = m_rabbits[m_nRabbits-1]; // fill in this position with the last rabbit in the array
                        m_rabbits[m_nRabbits-1] = nullptr;

                        m_nRabbits - -;
                }
        }
};
```