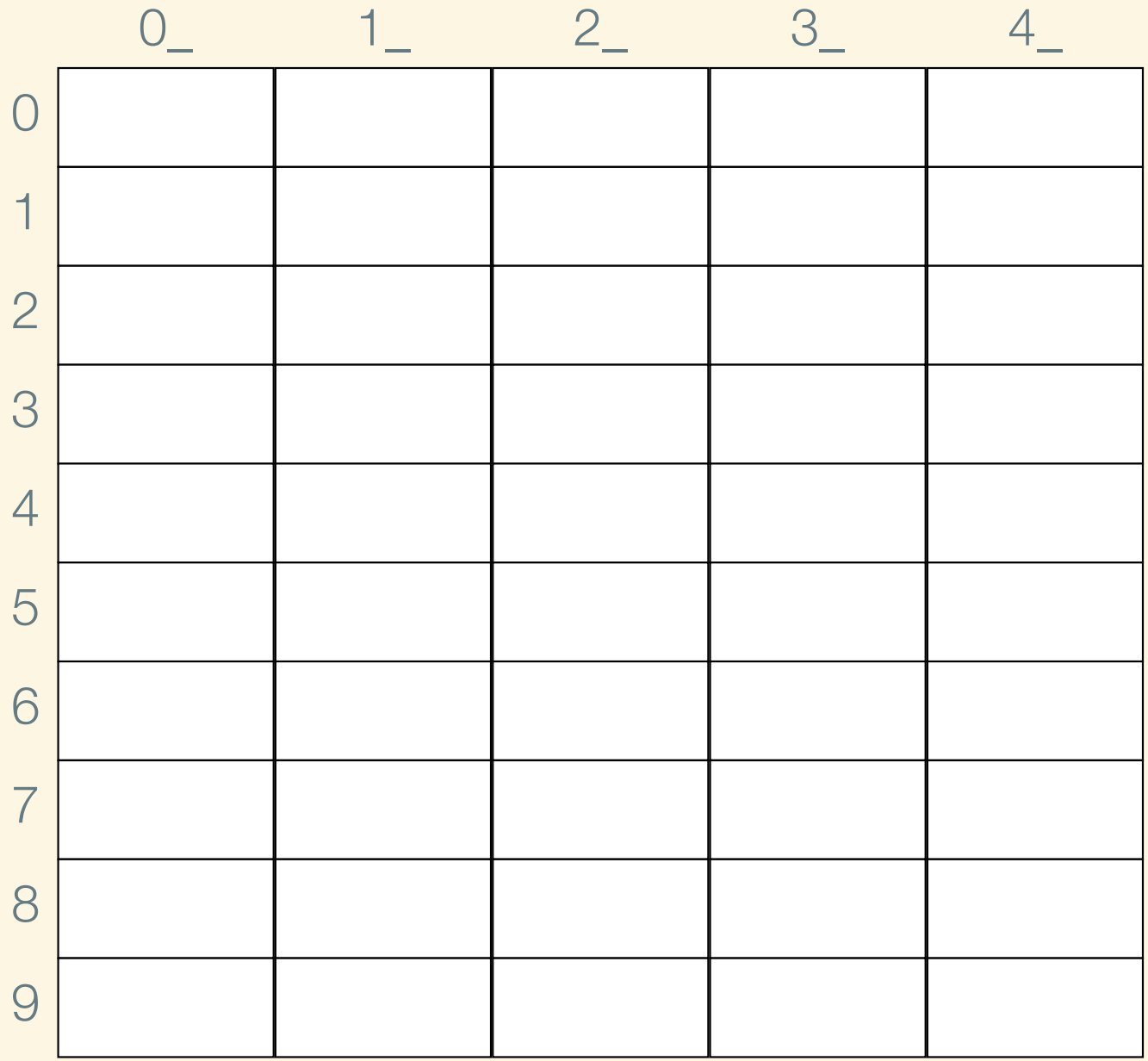
Pointers, Memory, and the Free Store

EECS 211

Winter 2017

00	10	20	30	40
01	11	21	31	41
02	12	22	32	42
03	13	23	33	43
04	14	24	34	44
05	15	25	35	45
06	16	26	36	46
07	17	27	37	47
08	18	28	38	48
09	19	29	39	49

	0_	1_	2_	3_	4_
0	00	10	20	30	40
1	01	11	21	31	41
2	02	12	22	32	42
3	03	13	23	33	43
4	04	14	24	34	44
5	05	15	25	35	45
6	06	16	26	36	46
7	07	17	27	37	47
8	08	18	28	38	48
9	09	19	29	39	49


0_
1_
2_
3_
4_
0
1
2
3
4
5
6
7
8
9

How to represent an aquarium?

Suppose you don't have std::vector (because someone had to write that *in* C++).

```
struct Fish { ··· };
```

How to represent an aquarium?

Suppose you don't have std::vector (because someone had to write that *in* C++).

```
struct Fish { ··· };

struct One_fish_aquarium { Fish fish_one; };
```

How to represent an aquarium?

Suppose you don't have std::vector (because someone had to write that *in* C++).

```
struct Fish { ··· };

struct One_fish_aquarium { Fish fish_one; };
struct Two_fish_aquarium { Fish fish_one, fish_two; };
```

How to represent an aquarium?

Suppose you don't have std::vector (because someone had to write that *in* C++).

```
struct Fish { ··· };

struct One_fish_aquarium { Fish fish_one; };
struct Two_fish_aquarium { Fish fish_one, fish_two; };
struct Three_fish_aquarium { Fish fish1, fish2; };
```

How to represent an aquarium?

Suppose you don't have std::vector (because someone had to write that *in* C++).

```
struct Fish { ··· };

struct One_fish_aquarium { Fish fish_one; };
struct Two_fish_aquarium { Fish fish_one, fish_two; };
struct Three_fish_aquarium { Fish fish1, fish2; };
struct Five_fish_aquarium { Fish f1, f2, f3, f4, f5; };
```

How to represent an aquarium?

Suppose you don't have std::vector (because someone had to write that *in* C++).

```
struct Fish { ··· };

struct One_fish_aquarium { Fish fish_one; };
struct Two_fish_aquarium { Fish fish_one, fish_two; };
struct Three_fish_aquarium { Fish fish1, fish2; };
struct Five_fish_aquarium { Fish f1, f2, f3, f4, f5; };

struct N_fish_aquarium
{
        Fish first;
        N_fish_aquarium rest;
};
```

Problem: C++ requires fixed sizes

Suppose that sizeof(Fish) == 8. Then:

Problem: C++ requires fixed sizes

Suppose that sizeof(Fish) == 8. Then:

```
struct One_fish_aquarium { Fish fish_one; };

sizeof(One_fish_aquarium) == 8
```

Problem: C++ requires fixed sizes

Suppose that sizeof(Fish) == 8. Then:

```
struct Two_fish_aquarium { Fish fish_one, fish_two; };

        sizeof(Two_fish_aquarium) == 16
```

Problem: C++ requires fixed sizes

Suppose that sizeof(Fish) == 8. Then:

```
struct Five_fish_aquarium { Fish f1, f2, f3, f4, f5; };

sizeof(Five_fish_aquarium) == 40
```

Problem: C++ requires fixed sizes

Suppose that sizeof(Fish) == 8. Then:

```
struct N_fish_aquarium
{
        Fish first;
        N_fish_aquarium rest;
};
```

sizeof(N_fish_aquarium) == 8 + sizeof(N_fish_aquarium) ???

Introducing the free store

Solution: put the *rest* somewhere else

Introducing the free store

Solution: put the *rest* somewhere else

Where? The *free store*!

Introducing the free store

Solution: put the *rest* somewhere else

Where? The *free store*! (A/k/a, *the heap*!)

Understanding memory

	0_	1_	2_
0	0	2	10
1	5	3	4
2	0	4	3
3	0	-1	0
4	-1	-1	5
5	-1	-1	12
6	-1	87	3
7	-1	4	4
8	66	16	10
9	0	-9	50

```
void f() {
```

Understanding memory

	0_	1_	2_
0	0	2	10
1	5	3	4
2	0	4	3
3	0	-1	0
4	-1	-1	5
5	-1	-1	12
6	-1	87	3
7	-1	4	4
8	66	16	10
9	0	-9	50

```
void f() {
    int x = 50;
```

Understanding memory

	0_	1_	2_
0	0	2	10
1	5	3	4
2	0	4	3
3	0	-1	0
4	-1	-1	5
5	-1	-1	12
6	-1	87	3
7	-1	4	4
8	66	16	10
9	0	-9	50

```
void f() {
    int x = 50;
    // int x @ 29
```

Understanding memory

	0_	1_	2_
0	0	2	10
1	5	3	4
2	0	4	3
3	0	-1	0
4	-1	-1	5
5	-1	-1	12
6	-1	87	3
7	-1	4	4
8	66	16	10
9	0	-9	50

```
void f() {
    int x = 50;
    // int x @ 29

    int y = 10;
```

Understanding memory

	0_	1_	2_
0	0	2	10
1	5	3	4
2	0	4	3
3	0	-1	0
4	-1	-1	5
5	-1	-1	12
6	-1	87	3
7	-1	4	4
8	66	16	10
9	0	-9	50

```
void f() {
    int x = 50;
    // int x @ 29

    int y = 10;
    // int y @ 28
```

Understanding memory

	0_	1_	2_
0	0	2	10
1	5	3	4
2	0	4	3
3	0	-1	0
4	-1	-1	5
5	-1	-1	12
6	-1	87	3
7	-1	4	4
8	66	16	10
9	0	-9	50

```
void f() {
    int x = 50;
    // int x @ 29

    int y = 10;
    // int y @ 28

    struct Posn { int x, y; }
```

Understanding memory

	0_	1_	2_
0	0	2	10
1	5	3	4
2	0	4	3
3	0	−1	0
4	−1	−1	5
5	−1	−1	12
6	−1	87	3
7	−1	4	4
8	66	16	10
9	0	−9	50

```
void f() {
    int x = 50;
    // int x @ 29

    int y = 10;
    // int y @ 28

    struct Posn { int x, y; }

    Posn p1{3, 4}, p2{5, 12};
```

Understanding memory

	0_	1_	2_
0	0	2	10
1	5	3	4
2	0	4	3
3	0	−1	0
4	−1	−1	5
5	−1	−1	12
6	−1	87	3
7	−1	4	4
8	66	16	10
9	0	−9	50

```
void f() {
    int x = 50;
    // int x @ 29

    int y = 10;
    // int y @ 28

    struct Posn { int x, y; }

    Posn p1{3, 4}, p2{5, 12};
    // Posn p1 @ 26, p2 @ 24
```

Understanding memory

	0_	1_	2_
0	0	2	10
1	5	3	4
2	0	4	3
3	0	−1	0
4	−1	−1	5
5	−1	−1	12
6	−1	87	3
7	−1	4	4
8	66	16	10
9	0	−9	50

```
void f() {
    int x = 50;
    // int x @ 29

    int y = 10;
    // int y @ 28

    struct Posn { int x, y; }

    Posn p1{3, 4}, p2{5, 12};
    // Posn p1 @ 26, p2 @ 24

    vector<int> v{2, 3, 4};
```

Understanding memory

	0_	1_	2_
0	0	2	10
1	5	3	4
2	0	4	3
3	0	−1	0
4	−1	−1	5
5	−1	−1	12
6	−1	87	3
7	−1	4	4
8	66	16	10
9	0	−9	50

```
void f() {
    int x = 50;
    // int x @ 29

    int y = 10;
    // int y @ 28

    struct Posn { int x, y; }

    Posn p1{3, 4}, p2{5, 12};
    // Posn p1 @ 26, p2 @ 24

    vector<int> v{2, 3, 4};
    // vector<int> v @ 20
```

Understanding memory

	0_	1_	2_
0	0	2	10
1	5	3	4
2	0	4	3
3	0	−1	0
4	−1	−1	5
5	−1	−1	12
6	−1	87	3
7	−1	4	4
8	66	16	10
9	0	−9	50

```
void f() {
    int x = 50;
    // int x @ 29

    int y = 10;
    // int y @ 28

    struct Posn { int x, y; }

    Posn p1{3, 4}, p2{5, 12};
    // Posn p1 @ 26, p2 @ 24

    vector<int> v{2, 3, 4};
    // vector<int> v @ 20
```

Understanding memory

	0_	1_	2_
0	0	2	10
1	5	3	4
2	0	4	4
3	0	5	0
4	−1	−1	5
5	−1	−1	12
6	−1	87	3
7	−1	4	4
8	66	16	10
9	0	−9	50

```
void f() {
    int x = 50;
    // int x @ 29

    int y = 10;
    // int y @ 28

    struct Posn { int x, y; }

    Posn p1{3, 4}, p2{5, 12};
    // Posn p1 @ 26, p2 @ 24

    vector<int> v{2, 3, 4};
    // vector<int> v @ 20

    v.push_back(5);
```

Understanding memory

	0_	1_	2_
0	0	2	2
1	5	3	8
2	2	4	5
3	3	5	0
4	4	−1	5
5	5	−1	12
6	6	87	3
7	−1	4	4
8	66	16	10
9	0	−9	50

```
void f() {
    int x = 50;
    // int x @ 29

    int y = 10;
    // int y @ 28

    struct Posn { int x, y; }

    Posn p1{3, 4}, p2{5, 12};
    // Posn p1 @ 26, p2 @ 24

    vector<int> v{2, 3, 4};
    // vector<int> v @ 20

    v.push_back(5);
    v.push_back(6);
```

Understanding memory

	0_	1_	2_
0	0	2	2
1	5	3	84
2	2	4	43
3	3	5	0
4	4	-1	5
5	1	-1	12
6	6	87	3
7	-1	4	4
8	66	16	10
9	0	-9	50

```
#include <memory>
```

Understanding memory

	0_	1_	2_
0	0	2	2
1	5	3	84
2	1	4	43
3	5	5	0
4	4	-1	5
5	1	-1	12
6	6	87	3
7	-1	4	4
8	66	16	10
9	0	-9	2

```
#include <memory>

std::shared_ptr<int> px =
        std::make_shared<int>(5);
```

Understanding memory

	0_	1_	2_
0	0	2	2
1	5	3	84
2	1	4	43
3	5	5	0
4	1	−1	5
5	7	−1	12
6	6	87	3
7	−1	4	4
8	66	16	4
9	0	−9	2

```
#include <memory>

std::shared_ptr<int> px =
    std::make_shared<int>(5);

std::shared_ptr<int> py =
    std::make_shared<int>(7);
```

Understanding memory

	0_	1_	2_
0	0	2	2
1	5	3	84
2	2	4	43
3	5	5	0
4	1	−1	5
5	7	−1	12
6	6	87	3
7	−1	4	2
8	66	16	4
9	0	−9	2

```
#include <memory>

std::shared_ptr<int> px =
      std::make_shared<int>(5);

std::shared_ptr<int> py =
      std::make_shared<int>(7);

std::shared_ptr<int> pz = px;
```

Understanding memory

	0_	1_	2_
0	0	2	2
1	5	3	84
2	2	4	43
3	6	5	0
4	1	−1	5
5	7	−1	12
6	6	87	3
7	−1	4	2
8	66	16	4
9	0	−9	2

```
#include <memory>

std::shared_ptr<int> px =
      std::make_shared<int>(5);

std::shared_ptr<int> py =
      std::make_shared<int>(7);

std::shared_ptr<int> pz = px;

*pz = 6;
```

Understanding memory

	0_	1_	2_
0	0	2	2
1	5	3	84
2	3	4	43
3	6	5	0
4	0	−1	5
5	7	−1	12
6	6	87	3
7	−1	4	2
8	66	16	2
9	0	−9	2

```
#include <memory>

std::shared_ptr<int> px =
      std::make_shared<int>(5);

std::shared_ptr<int> py =
      std::make_shared<int>(7);

std::shared_ptr<int> pz = px;

*pz = 6;

py = pz;
```

Understanding memory

	0_	1_	2_
0	0	2	2
1	5	3	84
2	2	4	43
3	6	5	0
4	1	−1	5
5	12	−1	12
6	6	87	3
7	−1	4	2
8	66	16	2
9	0	−9	4

```
#include <memory>

std::shared_ptr<int> px =
      std::make_shared<int>(5);

std::shared_ptr<int> py =
      std::make_shared<int>(7);

std::shared_ptr<int> pz = px;

*pz = 6;

py = pz;

px = std::make_shared<int>(12);
```

How to use shared pointers

The type of a shared pointer to a T is std::shared_ptr<T>.

How to use shared pointers

The type of a shared pointer to a T is std::shared_ptr<T>.

To create a shared pointer to a T, use std::make_shared<T>():

```
std::shared_ptr<double> p = std::make_shared<double>(5.5);
```

How to use shared pointers

The type of a shared pointer to a T is std::shared_ptr<T>.

To create a shared pointer to a T, use std::make_shared<T>():

```
std::shared_ptr<double> p = std::make_shared<double>(5.5);
```

Dereference a shared pointer with the prefix * operator:

```
std::cout << *p << '  ' << sqrt(*p) << '\n';
```

Understanding memory

	0_	1_	2_
0	0	2	2
1	5	3	84
2	2	4	43
3	5	5	0
4	6	-1	5
5	-1	-1	-8
6	0	87	3
7	7	4	17
8	66	16	14
9	0	-9	9

```
struct Cell
    { Fish first; Aquarium rest; };

using Aquarium =
        std::shared_ptr<Cell>;
```

Understanding memory

	0_	1_	2_
0	0	2	2
1	5	3	84
2	<><	4	43
3	∅	5	0
4	6	−1	5
5	−1	−1	−8
6	0	87	3
7	7	4	17
8	66	16	14
9	0	−9	2

```
struct Cell
    { Fish first; Aquarium rest; };

using Aquarium =
        std::shared_ptr<Cell>;

Aquarium a =
        std::make_shared<Cell>();
```

Understanding memory

	0_	1_	2_
0	0	2	2
1	5	3	84
2	<><	4	43
3	∅	5	0
4	6	−1	5
5	−1	−1	−8
6	<><	87	3
7	∅	4	17
8	66	16	6
9	0	−9	2

```
struct Cell
    { Fish first; Aquarium rest; };

using Aquarium =
        std::shared_ptr<Cell>;

Aquarium a =
        std::make_shared<Cell>();
Aquarium b =
        std::make_shared<Cell>();
```

Understanding memory

	0_	1_	2_
0	0	2	2
1	5	3	84
2	<><	4	43
3	∅	5	0
4	<><	−1	5
5	∅	−1	−8
6	<><	87	3
7	∅	4	4
8	66	16	6
9	0	−9	2

```
struct Cell
    { Fish first; Aquarium rest; };

using Aquarium =
        std::shared_ptr<Cell>;

Aquarium a =
        std::make_shared<Cell>();
Aquarium b =
        std::make_shared<Cell>();
Aquarium c =
        std::make_shared<Cell>();
```

Understanding memory

	0_	1_	2_
0	0	2	2
1	5	3	84
2	<><	4	43
3	∅	5	0
4	<><	−1	5
5	∅	−1	−8
6	<><	87	8
7	∅	4	4
8	<><	16	6
9	∅	−9	2

```
struct Cell
    { Fish first; Aquarium rest; };

using Aquarium =
        std::shared_ptr<Cell>;

Aquarium a =
        std::make_shared<Cell>();
Aquarium b =
        std::make_shared<Cell>();
Aquarium c =
        std::make_shared<Cell>();
Aquarium d =
        std::make_shared<Cell>();
```

Understanding memory

	0_	1_	2_
0	0	2	2
1	5	3	84
2	<><	4	43
3	6	5	0
4	<><	−1	5
5	∅	−1	−8
6	<><	87	8
7	∅	4	4
8	<><	16	6
9	∅	−9	2

```
struct Cell
    { Fish first; Aquarium rest; };

using Aquarium =
        std::shared_ptr<Cell>;

Aquarium a =
        std::make_shared<Cell>();
Aquarium b =
        std::make_shared<Cell>();
Aquarium c =
        std::make_shared<Cell>();
Aquarium d =
        std::make_shared<Cell>();

(*a).rest = b;
```

Understanding memory

	0_	1_	2_
0	0	2	2
1	5	3	84
2	<><	4	43
3	6	5	0
4	<><	−1	5
5	∅	−1	−8
6	<><	87	8
7	4	4	4
8	<><	16	6
9	∅	−9	2

```
struct Cell
    { Fish first; Aquarium rest; };

using Aquarium =
        std::shared_ptr<Cell>;

Aquarium a =
        std::make_shared<Cell>();
Aquarium b =
        std::make_shared<Cell>();
Aquarium c =
        std::make_shared<Cell>();
Aquarium d =
        std::make_shared<Cell>();

(*a).rest = b;
(*b).rest = c;
```

Understanding memory

	0_	1_	2_
0	0	2	2
1	5	3	84
2	<><	4	43
3	6	5	0
4	<><	−1	5
5	8	−1	−8
6	<><	87	8
7	4	4	4
8	<><	16	6
9	∅	−9	2

```
struct Cell
    { Fish first; Aquarium rest; };

using Aquarium =
        std::shared_ptr<Cell>;

Aquarium a =
        std::make_shared<Cell>();
Aquarium b =
        std::make_shared<Cell>();
Aquarium c =
        std::make_shared<Cell>();
Aquarium d =
        std::make_shared<Cell>();

(*a).rest = b;
(*b).rest = c;
(*c).rest = d;
```

– To CLion! –