

Sequence.h

```
class sequence
```

```
{
```

```
public:
```

```
int x;
```

```
void fun1()
```

```
{
```

```
fun2();
```

```
y = 20;
```

```
}
```

```
private:
```

```
int y;
```

```
void fun2();
```

```
}
```

```
void sequence::fun2()
```

```
{
```

```
x = y;
```

```
}
```

main

```
sequence s;
```

```
s.x = 10;
```

```
s.fun1();
```

```
s.y = 20; error
```

```
s.fun2(); error
```

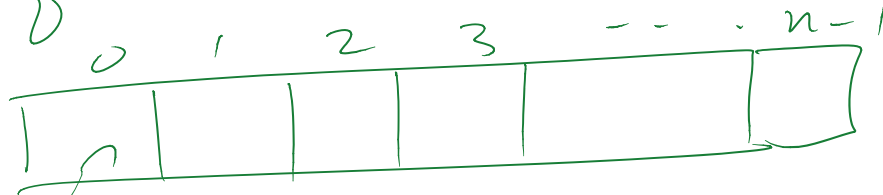
lecture 4

```
#include "sequence.h"
```

class structure

dynamic memory allocation

sequence



object of type T

}

```
int x;
int * p;
```

```
p = &x;
```

```
x = 20;
```

```
*p = 30;
```

← automatic storage

x: [~~30~~ 20] ←

p: [] → [50]

p = new int;

*p = 50;

allocate space

heap

A diagram showing a box labeled '500' representing a memory allocation on the heap. The box is outlined in blue and has a red arrow pointing to it from the word 'heap'.

stack

A diagram showing a stack of memory. It contains two entries: 'p: 500' and 'x: 200'. Below 'x: 200' is the value '100'. A red arrow points from the word 'stack' to the entries.

memory leak

```
p = int int [10];
```

```
*p = -100;
```

```
x(10+1) = -200;
```

A diagram of a memory stack. It shows a vertical list of indices 0, 1, 2, ..., 9. Next to index 0 is the value -100. Next to index 1 is the value -200. Next to index 2 is the value -200. A red arrow points from the word 'stack' to the list.

old school

$\ast(p+1) = -200;$

$\ast(p-1) = -300;$

200	p:
100	x:

$p[0] = -100;$

$p[1] = -200;$

$p[-1] = -300;$

$\{$ int A[10];

int \ast B;

B = new int[5];

A:

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

B:

0	1	2	3	4
1	1	1	1	1

B[] \rightarrow [1 1 1 1 1]

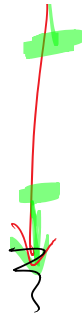
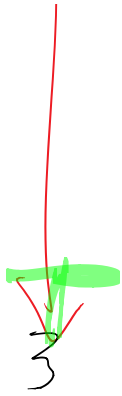
delete[] B;

$\}$ — A released

$\{$ int x;

$\{$ static int x;

$\{$ int x;



```
{ int x;  
  x = 10;  
}
```