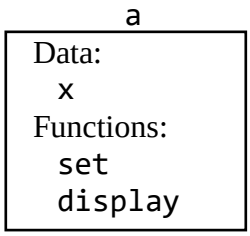
15 C++ Objects

Introduction

```
A a;  // creates an object named a from the class A
```

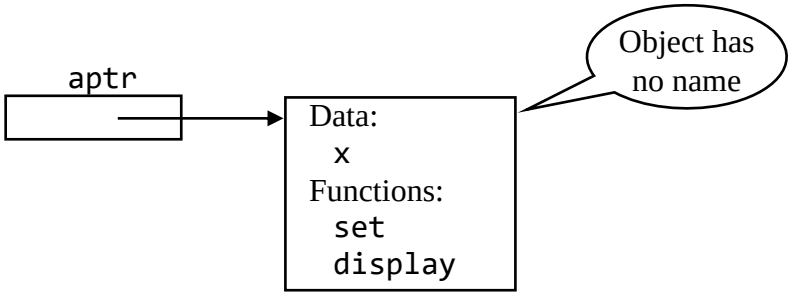



```
a.set(5);
a.x = 10;
```

Object itself has a name (“a”).

Dynamically Create A Object

```
A *aptr;
aptr = new A;
```




```
aptr->set(5);
aptr->x = 20;
```

Structs Versus Objects

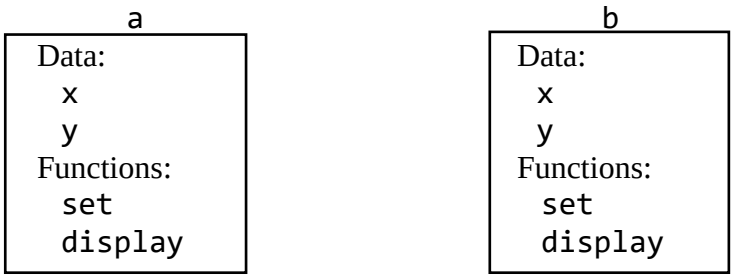
```
; ex1501.a  Structs in C++
startup   bl main
          halt
;==============================================================
                           ; #include <iostream>
                           ; using namespace std;
                           ; struct A
                           ; {
                           ;    int x;
                           ;    int y;
                           ; };
@set$p1Aii
          push lr          ; void set(A *r, int n, int m)
          push fp          ; {
          mov fp, sp

          ldr r0, fp, 3    ;    r->x = n;
          ldr r1, fp  2
          str r0, r1, 0

          ldr r0, fp, 4    ;    r->y = m;
          ldr r1, fp  2
          str r0, r1, 1

          mov sp, fp       ; }
          pop fp
          pop lr
          ret
;==============================================================
@display$p1A               ; void display(A *r)
          push lr          ; {
          push fp
          mov fp, sp

          ldr r0, fp, 2   ;    cout << r->x << endl;
          ldr r0, r0, 0
          dout r0
          nl

          ldr r0, fp, 2   ;    cout << r->y << endl;
          ldr r0, r0, 1
          dout r0
          nl

          mov sp, fp       ; }
          pop fp
          pop lr
          ret
```

```
;==============================================================
main      push lr           ; int main()
          push fp           ; {
          mov fp, sp

          sub sp, sp, 2     ;    A a, b;
          sub sp, sp, 2

          mov r0, 6         ;    set(&a, 5, 6);
          push r0
          mov r0, 5
          push r0
          add r0, fp, -2
          push r0
          bl @set$p1Aii
          add sp, sp, 3

          add r0, fp, -2    ;    display(&a);
          push r0
          bl @display$p1A
          add sp, sp, 1

          mov r0, 11        ;    set(&b, 10, 11);
          push r0
          mov r0, 10
          push r0
          add r0, fp, -4
          push r0
          bl @set$p1Aii
          add sp, sp, 3

          add r0, fp, -4    ;    display(&b);
          push r0
          bl @display$p1A
          add sp, sp, 1

          mov r0, 0         ;    return 0;
          mov sp, fp
          pop fp
          pop lr
          ret
                            ; }
```

Equivalent C++ Program Using Objects

```
 1 // ex1502.cpp  Objects in C++
 2 #include <iostream>
 3 using namespace std;
 4 class A
 5 {
 6    public:
 7       void set(int n, int m);
 8       void display();
 9    private:
10       int x;
11       int y;
12 };
13 void A::set(int n, int m)
14 {
15    x = n;
16    y = m;
17 }
18 void A::display()
19 {
20    cout << x << endl;
21    cout << y << endl;
22 }
23 //===================
24 int main()
25 {
26    A a, b;
27    a.set(5, 6);
28    a.display();
29    b.set(10, 11);
30    b.display();
31 // b.x = 20;          illegal because x is private
32    return 0;
33 }
```

Conceptual Picture



Two questions:

1. Each object has its own `set` and `display` functions. Is this not a very inefficient use of memory—to have multiple copies of the `set` and `display` functions?

2. The `set` functions in the `a` and `b` objects *are identical*. How then can the `set` functions have a different effect? For example, the call of the `set` function on line 27

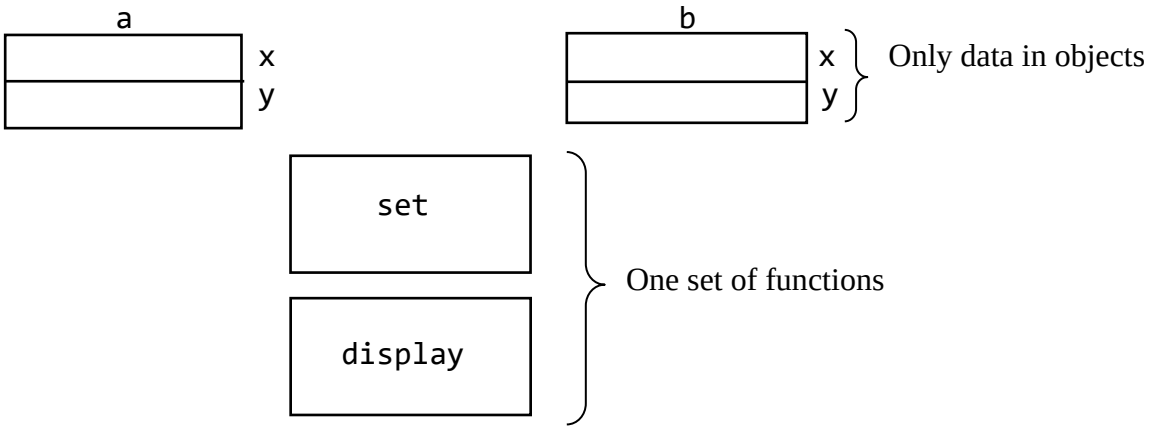
   ```
   a.set(5, 6);
   ```

   initializes the `x` and `y` *in the* `a` *object*, but the call of the *identical* `set` function on line 29

   ```
   b.set(10, 11);
   ```

   initializes the `x` and `y` *in the* `b` *object*.

Actual Picture



```
; ex1502.a  Objects in C++
startup   bl main
          halt
;==============================================================
                           ; #include <iostream>
                           ; using namespace std;
                           ; class A
                           ; {
                           ;    public:
                           ;       void set(int n);
                           ;       void display();
                           ;    private:
                           ;       int x;
                           ;       int y;
                           ; };

@A@set$ii
          push lr          ; void A::set(int n, int m)
          push fp          ; {
          mov fp, sp

          ldr r0, fp, 3    ;    x = n;
          ldr r1, fp, 2
          str r0, r1, 0

          ldr r0, fp, 4    ;    y = m;
          ldr r1, fp, 2
          str r0, r1, 1

          mov sp, fp       ; }
          pop fp
          pop lr
          ret

@A@display$v               ; void A::display()
          push lr          ; {
          push fp
          mov fp, sp

          ldr r0, fp, 2    ;    cout << x << endl;
          ldr r0, r0, 0
          dout r0
          nl

          ldr r0, fp, 2    ;    cout << y << endl;
          ldr r0, r0, 1
          dout r0
          nl

          mov sp, fp       ; }
          pop fp
```

```
52          pop lr
53          ret
54 ;==============================================================
55 main     push lr           ; int main()
56          push fp           ; {
57          mov fp, sp
58
59          add sp, sp, -2    ;    A a, b;
60          add sp, sp, -2
61
62          mov r0, 6         ;    a.set(5, 6);
63          push r0
64          mov r0, 5
65          push r0
66          add r0, fp, -2
67          push r0
68          bl @A@set$ii
69          add sp, sp, 3
70
71          add r0, fp, -2    ;    a.display();
72          push r0
73          bl @A@display$v
74          add sp, sp, 1
75
76          mov r0, 11         ;    b.set(10, 11);
77          push r0
78          mov r0, 10
79          push r0
80          add r0, fp, -4
81          push r0
82          bl @A@set$ii
83          add sp, sp, 3
84
85          add r0, fp, -4    ;    b.display();
86          push r0
87          bl @A@display$v
88          add sp, sp, 1
89
90          mov r0, 0         ;    return 0;
91          mov sp, fp
92          pop fp
93          pop lr
94          ret
95                            ; }
```

Get address of a

Get address of a

Get address of b

Get address of b

Creating Structs with malloc and Objects with new

To dynamically allocate the structs, we use

Casts the pointer returned by `malloc` to `A *`

```
p = (A *)malloc(sizeof(A));
q = (A *)malloc(sizeof(A));
```

Code for object version	Struct version	Object version
`main push lr`	`int main()`	`int main()`
`  push fp`	`{`	`{`
`  mov fp, sp`		
`  sub sp, sp, 1`	`A *p, *q;`	`A *p, *q;`
`  sub sp, sp, 1`		
`  mov r1, 2`	`p = (A *)malloc(sizeof(A));`	`p = new A;`
`  bl malloc`		
`  str r0, fp, -1`		
`  mov r1, 2`	`q = (A *)malloc(sizeof(A));`	`q = new A;`
`  bl malloc`		
`  str r0, fp, -2`		
`  mov r0, 6`	`set(p, 5, 6);`	`p->set(5, 6);`
`  push r0`		
`  mov r0, 5`		
`  push r0`		
`  ldr r0, fp, -1`		
`  push r0`		
`  bl @A@set$ii`		
`  add sp, sp, 3`		
`  ldr r0, fp, -1`	`display(p)`	`p->display();`
`  push r0`		
`  bl @A@display$v`		
`  add sp, sp, 1`		
`  mov r0, 11`	`set(q, 10, 11);`	`q->set(10, 11);`
`  push r0`		
`  mov r0, 10`		
`  push r0`		
`  ldr r0, fp, -2`		
`  push r0`		
`  bl @A@set$ii`		
`  add sp, sp, 3`		
`  ldr r0, fp, -2`	`display(q);`	`q->display();`
`  push r0`		
`  bl @A@display$v`		
`  add sp, sp, 1`		
`  mov r0, 0`	`return 0;`	`return 0;`
`  mov sp, fp`		
`  pop fp`		
`  pop lr`		
`  ret`		
	`}`	`}`
`malloc ld r0, @avail`		
`  add r1, r0, r1`		
`  st  r1, @avail`		
`  ret`		
`@avail .word @avail+1`		