

```
1. //
2. // main.c
3. // Array
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Demonstrates arrays.
10. //
11.
12. #include <stdio.h>
13.
14. int main(int argc, const char * argv[])
15. {
16.     // prompt user for number of exams
17.     int n;
18.     printf("Enter number of exams: ");
19.     scanf("%d", &n);
20.
21.     // allocate memory for grades (on stack)
22.     int grades[n];
23.
24.     // prompt user for exams' grades
25.     for (int i = 0; i < n; i++) {
26.         printf("Enter grade %d of %d: ", i+1, n);
27.         scanf("%d", &grades[i]);
28.     }
29.
30.     // do something with grades...
31.
32.     return 0;
33. }
```

```
1. //
2. // main.c
3. // Conditions
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Reports whether user's input is positive, negative, or zero.
10. //
11.
12. #include <stdio.h>
13.
14. int main(int argc, const char * argv[])
15. {
16.     int n;
17.     printf("Enter an integer: ");
18.     scanf("%d", &n);
19.     if (n > 0) {
20.         printf("Thanks for the positive integer!\n");
21.     }
22.     else if (n < 0) {
23.         printf("Thanks for the negative integer!\n");
24.     }
25.     else {
26.         printf("Thanks for the zero!\n");
27.     }
28.     return 0;
29. }
```

```
1. //
2. // main.c
3. // DoWhile
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Demonstrates a do-while loop.
10. //
11.
12. #include <stdio.h>
13.
14. int main(int argc, const char * argv[])
15. {
16.     int n;
17.     do {
18.         printf("Enter a positive integer: ");
19.         scanf("%d", &n);
20.     }
21.     while (n < 1);
22.     printf("Thanks for the positive integer!\n");
23.     return 0;
24. }
```

```
1. //
2. // main.c
3. // Enum
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Demonstrates an enum (and a struct).
10. //
11.
12. #include <stdio.h>
13.
14. // gender
15. typedef enum {
16.     FEMALE,
17.     MALE
18. } genders;
19.
20. // student
21. typedef struct {
22.     char *name;
23.     genders gender;
24. } student;
25.
26. // prototype
27. void greet(student s);
28.
29. int main(int argc, const char * argv[])
30. {
31.     // alice
32.     student alice;
33.     alice.name = "Alice";
34.     alice.gender = FEMALE;
35.     greet(alice);
36.
37.     // bob
38.     student bob;
39.     bob.name = "Bob";
40.     bob.gender = MALE;
41.     greet(bob);
42.
43.     return 0;
44. }
45.
46. // greets student
47. void greet(student s)
48. {
```

```
49.     char *title = (s.gender == FEMALE) ? "Ms." : "Mr.";
50.     printf("Hello, %s %s.\n", title, s.name);
51. }
```

```
1. //
2. // main.c
3. // For
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Demonstrates a for loop.
10. //
11.
12. #include <stdio.h>
13.
14. int main(int argc, const char * argv[])
15. {
16.     int n;
17.     printf("Enter a positive integer: ");
18.     scanf("%d", &n);
19.     for (int i = n; i > 0; i--) {
20.         printf("%d...\n", i);
21.     }
22.     printf("Blast off!\n");
23.     return 0;
24. }
```

```
1. //
2. // main.c
3. // GetInt
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Gets an int from the user.
10. //
11.
12. #include <stdio.h>
13.
14. int main(int argc, const char * argv[])
15. {
16.     int n;
17.     printf("Enter an integer: ");
18.     scanf("%d", &n);
19.     printf("Thanks for the %d!\n", n);
20.     return 0;
21. }
```

```
1. //
2. // main.c
3. // HelloC
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Says hello to the world in C.
10. //
11.
12. #include <stdio.h>
13.
14. int main (int argc, const char * argv[])
15. {
16.     int n = 50;
17.     printf("Hello, %s %d!\n", "World!", n);
18.     return 0;
19. }
```



```
1. //
2. // main.c
3. // Malloc
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Demonstrates malloc.
10. //
11.
12. #include <stdio.h>
13. #include <stdlib.h>
14.
15. // prototype
16. int *get_grades(int exams);
17.
18. int main (int argc, const char * argv[])
19. {
20.     // prompt user for number of exams
21.     int n;
22.     printf("Enter number of exams: ");
23.     scanf("%d", &n);
24.
25.     // get grades
26.     int *grades = get_grades(n);
27.
28.     // do something with grades...
29.
30.     // free memory
31.     free(grades);
32.
33.     return 0;
34. }
35.
36. // gets grades
37. int *get_grades(int exams)
38. {
39.     // allocate memory for grades (on heap)
40.     int *grades = malloc(sizeof(int) * exams);
41.
42.     // prompt user for exams' grades
43.     for (int i = 0; i < exams; i++) {
44.         printf("Enter grade %d of %d: ", i+1, exams);
45.         scanf("%d", &grades[i]);
46.     }
47.     return grades;
48. }
```

```
1. //
2. // main.c
3. // Struct
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Demonstrates a struct.
10. //
11.
12. #include <stdio.h>
13.
14. // student
15. typedef struct {
16.     int age;
17.     char *name;
18. } student;
19.
20. // prototype
21. void greet(student s);
22.
23. int main (int argc, const char * argv[])
24. {
25.     // alice
26.     student alice;
27.     alice.age = 20;
28.     alice.name = "Alice";
29.     greet(alice);
30.
31.     // bob
32.     student bob;
33.     bob.age = 21;
34.     bob.name = "Bob";
35.     greet(bob);
36.
37.     return 0;
38. }
39.
40. // greets student
41. void greet(student s)
42. {
43.     printf("Hello, %s. I see that you are %d years old.\n", s.name, s.age);
44. }
```

```
1. //
2. // main.c
3. // SwapFailure
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Fails to swap two variables' values.
10. //
11.
12. #include <stdio.h>
13.
14. // function prototype
15. void swap(int a, int b);
16.
17.
18. int main(int argc, const char * argv[])
19. {
20.     int x = 0;
21.     int y = 1;
22.
23.     printf("x is %d\n", x);
24.     printf("y is %d\n", y);
25.     printf("Swapping x and y...\n");
26.     swap(x, y);
27.     printf("Success!\n");
28.     printf("x is %d\n", x);
29.     printf("y is %d\n", y);
30.
31.     return 0;
32. }
33.
34.
35. //
36. // Swaps arguments' values.
37. //
38.
39. void swap(int a, int b)
40. {
41.     int tmp = a;
42.     a = b;
43.     b = tmp;
44. }
```

```
1. //
2. // main.c
3. // SwapSuccess
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Swaps two variables' values.
10. //
11.
12. #include <stdio.h>
13.
14. // function prototype
15. void swap(int *a, int *b);
16.
17.
18. int main(int argc, const char * argv[])
19. {
20.     int x = 0;
21.     int y = 1;
22.
23.     printf("x is %d\n", x);
24.     printf("y is %d\n", y);
25.     printf("Swapping x and y...\n");
26.     swap(&x, &y);
27.     printf("Success!\n");
28.     printf("x is %d\n", x);
29.     printf("y is %d\n", y);
30.
31.     return 0;
32. }
33.
34.
35. //
36. // Swaps arguments' values.
37. //
38.
39. void swap(int *a, int *b)
40. {
41.     int tmp = *a;
42.     *a = *b;
43.     *b = tmp;
44. }
```

```
1. //
2. // main.c
3. // While
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Demonstrates a while loop.
10. //
11.
12. #include <stdio.h>
13.
14. int main(int argc, const char * argv[])
15. {
16.     int n;
17.     printf("Enter a positive integer: ");
18.     scanf("%d", &n);
19.     while (n > 0) {
20.         printf("%d...\n", n);
21.         n--;
22.     }
23.     printf("Blast off!\n");
24.     return 0;
25. }
```

```
1. //
2. // main.c
3. // HelloObjC
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Says hello to the world in Objective-C.
10. //
11.
12. #import <Foundation/Foundation.h>
13.
14. int main(int argc, const char * argv[])
15. {
16.     @autoreleasepool {
17.         NSLog(@"Hello, World!");
18.     }
19.     return 0;
20. }
```

```
1. //
2. // main.m
3. // Students1
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Demonstrates use of a class.
10. //
11.
12. #import <Foundation/Foundation.h>
13. #import "Student.h"
14.
15. // prototype
16. void greet(Student *s);
17.
18. int main(int argc, const char * argv[])
19. {
20.     @autoreleasepool {
21.
22.         // Alice
23.         Student *alice = [Student alloc];
24.         alice->age = 20;
25.         alice->name = @"Alice";
26.         greet(alice);
27.
28.         // Bob
29.         Student *bob = [Student alloc];
30.         bob->age = 21;
31.         bob->name = @"Bob";
32.         greet(bob);
33.
34.     }
35.     return 0;
36. }
37.
38. // greets student (via stderr)
39. void greet(Student *s)
40. {
41.     NSLog(@"Hello, %@. I see that you are %d years old.\n", s->name, s->age);
42. }
```

```
1. //
2. // Student.h
3. // Students1
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Declares a student.
10. //
11.
12. #import <Foundation/Foundation.h>
13.
14. @interface Student : NSObject {
15. @public
16.     int age;
17.     NSString *name;
18. }
19. @end
```



```
1. //
2. // Student.m
3. // Students1
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Defines a student.
10. //
11.
12. #import "Student.h"
13.
14. @implementation Student
15. @end
```

```
1. //
2. // main.m
3. // Students2
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Demonstrates use of a class, getters, and setters.
10. //
11.
12. #import <Foundation/Foundation.h>
13. #import "Student.h"
14.
15. // prototype
16. void greet(Student *s);
17.
18. int main(int argc, const char * argv[])
19. {
20.     @autoreleasepool {
21.
22.         // Alice
23.         Student *alice = [[Student alloc] init];
24.         [[alice setAge:20] setName:@"Alice"];
25.         greet(alice);
26.
27.         // Bob
28.         Student *bob = [[Student alloc] init];
29.         [bob setAge:21];
30.         [bob setName:@"Bob"];
31.         greet(bob);
32.     }
33.     return 0;
34. }
35.
36. // greets student (via stderr)
37. void greet(Student *s)
38. {
39.     NSLog(@"Hello, %@. I see that you are %d years old.\n", [s name], [s age]);
40. }
```

```
1. //
2. // Student.h
3. // Students2
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Declares a student with getters and setters.
10. //
11.
12. #import <Foundation/Foundation.h>
13.
14. @interface Student : NSObject {
15.     int _age;
16.     NSString *_name;
17. }
18.
19. - (int)age;
20. - (Student *)setAge:(int)age;
21.
22. - (NSString *)name;
23. - (Student *)setName:(NSString *)name;
24.
25. @end
```

```
1. //
2. // Student.m
3. // Students2
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Defines a student with getters and setters.
10. //
11.
12. #import "Student.h"
13.
14. @implementation Student
15.
16. - (int)age
17. {
18.     return _age;
19. }
20.
21. - (Student *)setAge:(int)age
22. {
23.     _age = age;
24.     return self;
25. }
26.
27. - (NSString *)name
28. {
29.     return _name;
30. }
31.
32. - (Student *)setName:(NSString *)name
33. {
34.     _name = [name copy];
35.     return self;
36. }
37.
38. @end
```

```
1. //
2. // main.m
3. // Students3
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Demonstrates use of properties.
10. //
11.
12. #import <Foundation/Foundation.h>
13. #import "Student.h"
14.
15. // prototype
16. void greet(Student *s);
17.
18. int main(int argc, const char * argv[])
19. {
20.     @autoreleasepool {
21.
22.         // Alice
23.         Student *alice = [[Student alloc] init];
24.         alice.age = 20;
25.         alice.name = @"Alice";
26.         greet(alice);
27.
28.         // Bob
29.         Student *bob = [[Student alloc] init];
30.         bob.age = 21;
31.         bob.name = @"Bob";
32.         greet(bob);
33.     }
34.     return 0;
35. }
36.
37. // greets student (via stderr)
38. void greet(Student *s)
39. {
40.     NSLog(@"Hello, %@. I see that you are %d years old.\n", s.name, s.age);
41. }
```

```
1. //
2. // Student.h
3. // Students3
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Declares a student with properties.
10. //
11.
12. #import <Foundation/Foundation.h>
13.
14. @interface Student : NSObject {
15.     int _age;
16.     NSString *_name;
17. }
18.
19. @property (assign, nonatomic, readwrite) int age;
20. @property (copy, nonatomic, readwrite) NSString *name;
21.
22. - (int)age;
23. - (void)setAge:(int)age;
24.
25. - (NSString *)name;
26. - (void)setName:(NSString *)name;
27.
28. @end
```

```
1. //
2. // Student.m
3. // Students3
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Defines a student with properties.
10. //
11.
12. #import "Student.h"
13.
14. @implementation Student
15.
16. - (int)age
17. {
18.     return _age;
19. }
20.
21. - (void)setAge:(int)age
22. {
23.     _age = age;
24. }
25.
26. - (NSString *)name
27. {
28.     return _name;
29. }
30.
31. - (void)setName:(NSString *)name
32. {
33.     _name = [name copy];
34. }
35.
36. @end
```

```
1. //
2. // main.m
3. // Students4
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Demonstrates use of synthesized properties.
10. //
11.
12. #import <Foundation/Foundation.h>
13. #import "Student.h"
14.
15. // prototype
16. void greet(Student *s);
17.
18. int main(int argc, const char * argv[])
19. {
20.     @autoreleasepool {
21.
22.         // Alice
23.         Student *alice = [[Student alloc] init];
24.         alice.age = 20;
25.         alice.name = @"Alice";
26.         greet(alice);
27.
28.         // Bob
29.         Student *bob = [[Student alloc] init];
30.         bob.age = 21;
31.         bob.name = @"Bob";
32.         greet(bob);
33.     }
34.     return 0;
35. }
36.
37. // greets student (via stderr)
38. void greet(Student *s)
39. {
40.     NSLog(@"Hello, %@. I see that you are %d years old.\n", s.name, s.age);
41. }
```



```
1. //
2. // Student.h
3. // Students4
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Declares a student with synthesized properties.
10. //
11.
12. #import <Foundation/Foundation.h>
13.
14. @interface Student : NSObject {
15. }
16.
17. @property (assign, nonatomic, readwrite) int age;
18. @property (copy, nonatomic, readwrite) NSString *name;
19.
20. @end
```

```
1. //
2. // Student.m
3. // Students4
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Defines a student with synthesized properties.
10. //
11.
12. #import "Student.h"
13.
14. @implementation Student
15. @end
```

```
1. //
2. // main.m
3. // Students5
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Demonstrates use of (mostly) synthesized properties.
10. //
11.
12. #import <Foundation/Foundation.h>
13. #import "Student.h"
14.
15. // prototype
16. void greet(Student *s);
17.
18. int main(int argc, const char * argv[])
19. {
20.     @autoreleasepool {
21.
22.         // Alice
23.         Student *alice = [[Student alloc] init];
24.         alice.age = 20;
25.         alice.name = @"Alice";
26.         greet(alice);
27.
28.         // Bob
29.         Student *bob = [[Student alloc] init];
30.         bob.age = 21;
31.         bob.name = @"Bob";
32.         greet(bob);
33.
34.         // David
35.         Student *david = [[Student alloc] init];
36.         david.age = 29;
37.         david.name = @"David";
38.         greet(david);
39.     }
40.     return 0;
41. }
42.
43. // greets student (via stderr)
44. void greet(Student *s)
45. {
46.     NSLog(@"Hello, %@. I see that you are %d years old.\n", s.name, s.age);
47. }
```

```
1. //
2. // Student.h
3. // Students5
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Declares a student with (mostly) synthesized properties.
10. //
11.
12. #import <Foundation/Foundation.h>
13.
14. @interface Student : NSObject {
15. }
16.
17. @property (assign, nonatomic, readwrite) int age;
18. @property (copy, nonatomic, readwrite) NSString *name;
19.
20. @end
```

```
1. //
2. // Student.m
3. // Students5
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Defines a student with (mostly) synthesized properties.
10. //
11.
12. #import "Student.h"
13.
14. @implementation Student
15.
16. - (void)setName:(NSString *)name
17. {
18.     if ([name isEqualToString:@"David"]) {
19.         _name = @"Dummy";
20.     }
21.     else {
22.         _name = [name copy];
23.     }
24. }
25.
26.
27. @end
```

```
1. //
2. // main.m
3. // Students6
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Demonstrates an init... method.
10. //
11.
12. #import <Foundation/Foundation.h>
13. #import "Student.h"
14.
15. // prototype
16. void greet(Student *s);
17.
18. int main(int argc, const char * argv[])
19. {
20.     @autoreleasepool {
21.
22.         // Alice
23.         Student *alice = [[Student alloc] initWithName:@"Alice" andAge:20];
24.         greet(alice);
25.
26.         // Bob
27.         Student *bob = [[Student alloc] initWithName:@"Bob" andAge:21];
28.         greet(bob);
29.
30.         // John
31.         Student *john = [[Student alloc] init];
32.         greet(john);
33.     }
34.     return 0;
35. }
36.
37. // greets student (via stderr)
38. void greet(Student *s)
39. {
40.     NSLog(@"Hello, %@. I see that you are %d years old.\n", s.name, s.age);
41. }
```

```
1. //
2. // Student.h
3. // Students6
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Declares a student with an init... method.
10. //
11.
12. #import <Foundation/Foundation.h>
13.
14. @interface Student : NSObject {
15. }
16.
17. @property (assign, nonatomic, readwrite) int age;
18. @property (copy, nonatomic, readwrite) NSString *name;
19.
20. - (id)initWithName:(NSString *)name andAge:(int)age;
21.
22. @end
```

```
1. //
2. // Student.m
3. // Students6
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Defines a student with an initWithName... method.
10. //
11.
12. #import "Student.h"
13.
14. @implementation Student
15.
16. - (id)init
17. {
18.     self = [self initWithName:@"John" andAge:406];
19.     return self;
20. }
21.
22. - (id)initWithName:(NSString *)name andAge:(int)age
23. {
24.     if (self = [super init])
25.     {
26.         self.age = age;
27.         self.name = name;
28.     }
29.     return self;
30. }
31.
32. @end
```



```
1. //
2. // main.m
3. // Students7
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Demonstrates mutable arrays.
10. //
11.
12. #import <Foundation/Foundation.h>
13. #import "Student.h"
14.
15. // prototype
16. void greet(Student *s);
17.
18. int main(int argc, const char * argv[])
19. {
20.     @autoreleasepool {
21.
22.         // allocate array for students
23.         NSMutableArray *students = [[NSMutableArray alloc] init];
24.
25.         // Alice
26.         Student *alice = [[Student alloc] initWithName:@"Alice" andAge:20];
27.         [students addObject:alice];
28.
29.         // Bob
30.         [students addObject:[[Student alloc] initWithName:@"Bob" andAge:21]];
31.
32.         // greet then release each student
33.         for (Student *s in students) {
34.             greet(s);
35.         }
36.     }
37.     return 0;
38. }
39.
40. // greets student (via stderr)
41. void greet(Student *s)
42. {
43.     NSLog(@"Hello, %@. I see that you are %d years old.\n", s.name, s.age);
44. }
```

```
1. //
2. // Student.h
3. // Students7
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Declares a student with init methods.
10. //
11.
12. #import <Foundation/Foundation.h>
13.
14. @interface Student : NSObject {
15. }
16.
17. @property (assign, nonatomic, readwrite) int age;
18. @property (copy, nonatomic, readwrite) NSString *name;
19.
20. - (id)initWithName:(NSString *)name andAge:(int)age;
21.
22. @end
```

```
1. //
2. // Student.m
3. // Students7
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Defines a student with init methods.
10. //
11.
12. #import "Student.h"
13.
14. @implementation Student
15.
16. - (id)init
17. {
18.     self = [self initWithName:@"John" andAge:406];
19.     return self;
20. }
21.
22. - (id)initWithName:(NSString *)name andAge:(int)age
23. {
24.     if (self = [super init])
25.     {
26.         self.age = age;
27.         self.name = name;
28.     }
29.     return self;
30. }
31.
32. @end
```

```
1. //
2. // main.m
3. // Students8
4. //
5. // Robert Bowden
6. // Harvard University
7. // rob@cs.harvard.edu
8. //
9. // Demonstrates sorting an array using a selector
10. //
11.
12. #import <Foundation/Foundation.h>
13. #import "Student.h"
14.
15. // prototype
16. void greet(Student *s);
17.
18. int main(int argc, const char * argv[])
19. {
20.     @autoreleasepool {
21.
22.         // allocate array for students
23.         NSMutableArray *students = [[NSMutableArray alloc] init];
24.
25.         // R.J.
26.         [students addObject:[Student alloc] initWithName:@"R.J." andAge:21];
27.
28.         // Chris
29.         [students addObject:[Student alloc] initWithName:@"Chris" andAge:18];
30.
31.         // Rob
32.         [students addObject:[Student alloc] initWithName:@"Rob" andAge:22];
33.
34.         // David
35.         [students addObject:[Student alloc] initWithName:@"David" andAge:85];
36.
37.         // sort the array
38.         NSArray *sortedStudents = [students sortedArrayUsingSelector:@selector(compare)];
39.
40.         // greet then release each student
41.         for (Student *s in sortedStudents) {
42.             greet(s);
43.         }
44.     }
45.     return 0;
46. }
47.
48. // greets student (via stderr)
```

```
49. void greet(Student *s)
50. {
51.     NSLog(@"Hello, %@. I see that you are %d years old.\n", s.name, s.age);
52. }
```

```
1. //
2. // Student.h
3. // Students8
4. //
5. // Robert Bowden
6. // Harvard University
7. // rob@cs.harvard.edu
8. //
9. // Declares a student with init methods.
10. //
11.
12. #import <Foundation/Foundation.h>
13.
14. @interface Student : NSObject {
15. }
16.
17. @property (assign, nonatomic, readwrite) int age;
18. @property (copy, nonatomic, readwrite) NSString *name;
19.
20. - (id)initWithName:(NSString *)name andAge:(int)age;
21. - (NSComparisonResult)compare:(Student *)otherStudent;
22.
23.
24. @end
```

```
1. //
2. // Student.m
3. // Students8
4. //
5. // Robert Bowden
6. // Harvard University
7. // rob@cs.harvard.edu
8. //
9. // Defines a student with init methods.
10. //
11.
12. #import "Student.h"
13.
14. @implementation Student
15.
16. - (id)init
17. {
18.     self = [self initWithName:@"John" andAge:406];
19.     return self;
20. }
21.
22. - (id)initWithName:(NSString *)name andAge:(int)age
23. {
24.     if (self = [super init])
25.     {
26.         self.age = age;
27.         self.name = name;
28.     }
29.     return self;
30. }
31.
32. - (NSComparisonResult)compare:(Student *)otherStudent {
33.     return [self.name compare:otherStudent.name];
34. }
35.
36. @end
```

```
1. //
2. // main.m
3. // Students9
4. //
5. // Robert Bowden
6. // Harvard University
7. // rob@cs.harvard.edu
8. //
9. // Demonstrates sorting an array using a selector from a category
10. //
11.
12. #import <Foundation/Foundation.h>
13. #import "Student+StudentCompare.h"
14.
15. // prototype
16. void greet(Student *s);
17.
18. int main(int argc, const char * argv[])
19. {
20.     @autoreleasepool {
21.
22.         // allocate array for students
23.         NSMutableArray *students = [[NSMutableArray alloc] init];
24.
25.         // R.J.
26.         [students addObject:[Student alloc] initWithName:@"R.J." andAge:21];
27.
28.         // Chris
29.         [students addObject:[Student alloc] initWithName:@"Chris" andAge:18];
30.
31.         // Rob
32.         [students addObject:[Student alloc] initWithName:@"Rob" andAge:22];
33.
34.         // David
35.         [students addObject:[Student alloc] initWithName:@"David" andAge:85];
36.
37.         // sort the array
38.         NSArray *sortedStudents = [students sortedArrayUsingSelector:@selector(compare)];
39.
40.         // greet then release each student
41.         for (Student *s in sortedStudents) {
42.             greet(s);
43.         }
44.     }
45.     return 0;
46. }
47.
48. // greets student (via stderr)
```



```
49. void greet(Student *s)
50. {
51.     NSLog(@"Hello, %@. I see that you are %d years old.\n", s.name, s.age);
52. }
```

```
1. //
2. // Student+StudentCompare.h
3. // Students9
4. //
5. // Robert Bowden
6. // Harvard University
7. // rob@cs.harvard.edu
8. //
9. // Demonstrates sorting an array using a selector from a category
10. //
11.
12. #import "Student.h"
13.
14. @interface Student (StudentCompare)
15.
16. - (NSComparisonResult)compare:(Student *)otherStudent;
17.
18. @end
```

```
1. //
2. // Student+StudentCompare.m
3. // Students9
4. //
5. // Robert Bowden
6. // Harvard University
7. // rob@cs.harvard.edu
8. //
9. // Demonstrates sorting an array using a selector from a category
10. //
11.
12. #import "Student+StudentCompare.h"
13.
14. @implementation Student (StudentCompare)
15.
16. - (NSComparisonResult)compare:(Student *)otherStudent {
17.     return [self.name compare:otherStudent.name];
18. }
19.
20. @end
```

```
1. //
2. // Student.h
3. // Students9
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Declares a student with init methods.
10. //
11.
12. #import <Foundation/Foundation.h>
13.
14. @interface Student : NSObject {
15. }
16.
17. @property (assign, nonatomic, readwrite) int age;
18. @property (copy, nonatomic, readwrite) NSString *name;
19.
20. - (id)initWithName:(NSString *)name andAge:(int)age;
21.
22. @end
```

```
1. //
2. // Student.m
3. // Students9
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Defines a student with init methods.
10. //
11.
12. #import "Student.h"
13.
14. @implementation Student
15.
16. - (id)init
17. {
18.     self = [self initWithName:@"John" andAge:406];
19.     return self;
20. }
21.
22. - (id)initWithName:(NSString *)name andAge:(int)age
23. {
24.     if (self = [super init])
25.     {
26.         self.age = age;
27.         self.name = name;
28.     }
29.     return self;
30. }
31.
32. @end
```

```
1. //
2. // main.m
3. // Students10
4. //
5. // Robert Bowden
6. // Harvard University
7. // rob@cs.harvard.edu
8. //
9. // Demonstrates sorting an array using a block
10. //
11.
12. #import <Foundation/Foundation.h>
13. #import "Student.h"
14.
15. // prototype
16. void greet(Student *s);
17.
18. int main(int argc, const char * argv[])
19. {
20.     @autoreleasepool {
21.
22.         // allocate array for students
23.         NSMutableArray *students = [[NSMutableArray alloc] init];
24.
25.         // R.J.
26.         [students addObject:[[Student alloc] initWithName:@"R.J." andAge:20]];
27.
28.         // Chris
29.         [students addObject:[[Student alloc] initWithName:@"Chris" andAge:20]];
30.
31.         // Rob
32.         [students addObject:[[Student alloc] initWithName:@"Rob" andAge:20]];
33.
34.         // David
35.         [students addObject:[[Student alloc] initWithName:@"David" andAge:21]];
36.
37.         // store the block in a variable called "func". Alternatively, could
38.         // inline the block in the line below, similar to anonymous functions
39.         // in JavaScript
40.         NSComparisonResult (^func)(Student *, Student *) = ^NSComparisonResult(Student *a, Student *b) {
41.             return [a.name compare:b.name];
42.         };
43.
44.         // sort the array using our block
45.         NSArray *sortedStudents = [students sortedArrayUsingComparator:func];
46.
47.         // greet then release each student
48.         for (Student *s in sortedStudents) {
```

```
49.         greet(s);
50.     }
51. }
52.     return 0;
53. }
54.
55. // greets student (via stderr)
56. void greet(Student *s)
57. {
58.     NSLog(@"Hello, %@. I see that you are %d years old.\n", s.name, s.age);
59. }
```

```
1. //
2. // Student.h
3. // Students10
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Declares a student with init methods.
10. //
11.
12. #import <Foundation/Foundation.h>
13.
14. @interface Student : NSObject {
15. }
16.
17. @property (assign, nonatomic, readwrite) int age;
18. @property (copy, nonatomic, readwrite) NSString *name;
19.
20. - (id)initWithName:(NSString *)name andAge:(int)age;
21.
22. @end
```



```
1. //
2. // Student.m
3. // Students10
4. //
5. // David J. Malan
6. // Harvard University
7. // malan@harvard.edu
8. //
9. // Defines a student with init methods.
10. //
11.
12. #import "Student.h"
13.
14. @implementation Student
15.
16. - (id)init
17. {
18.     self = [self initWithName:@"John" andAge:406];
19.     return self;
20. }
21.
22. - (id)initWithName:(NSString *)name andAge:(int)age
23. {
24.     if (self = [super init])
25.     {
26.         self.age = age;
27.         self.name = name;
28.     }
29.     return self;
30. }
31.
32. @end
```