Smith College

Computer Science

CSC231—C Tutorials

Fall 2017

Introduction to C

Dominique Thiébaut
dthiebaut@smith.edu

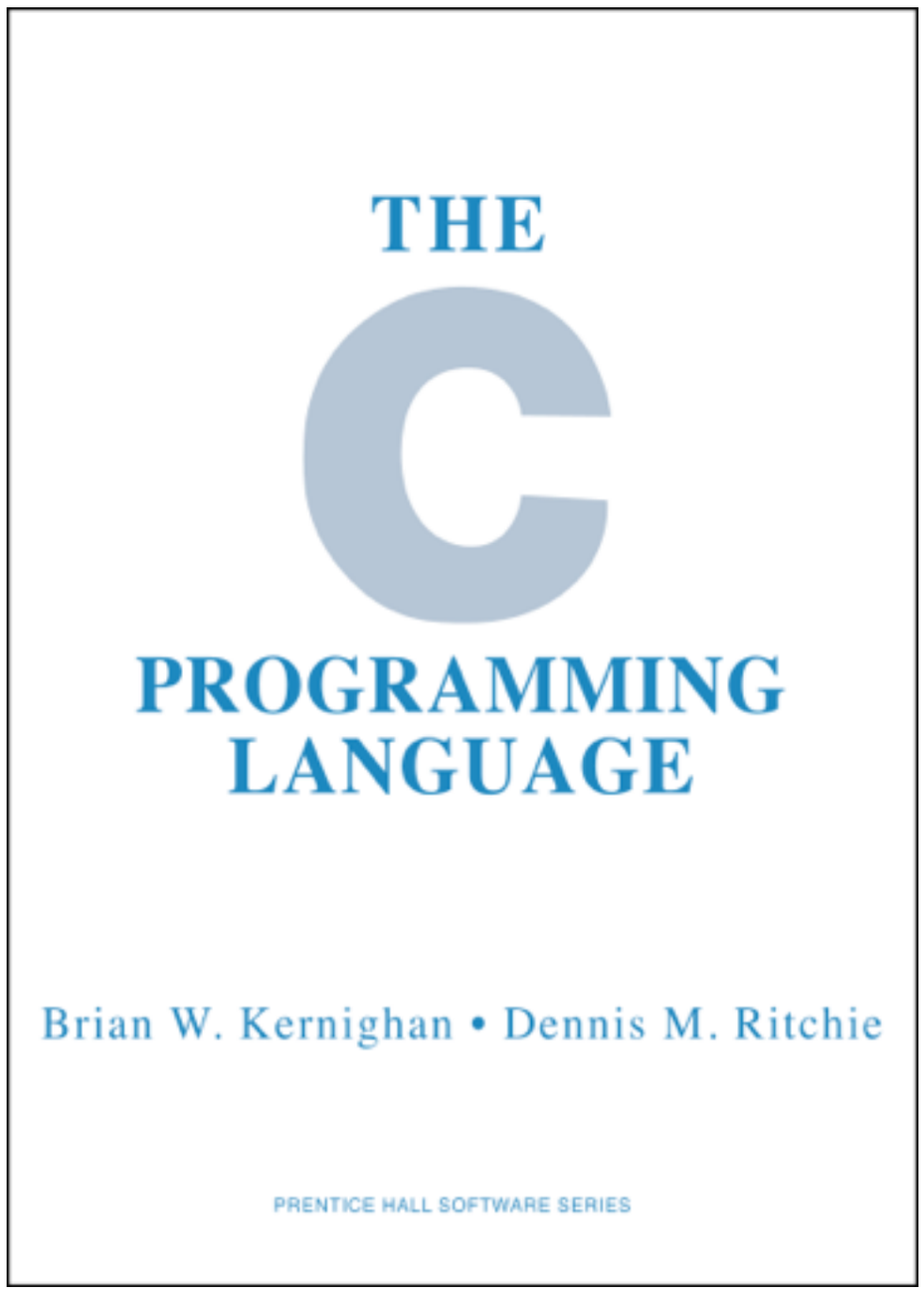
Smith College

Computer Science

Learning C in 4 Hours!

D. Thiebaut

Dominique Thiébaut
dthiebaut@smith.edu



- Dennis Ritchie
- 1969 to 1973
- AT&T Bell Labs
- Close to Assembly
- Unix

- Standard
- Many languages based on C. (C++, Obj. C, C#)
- Many influenced by C (Java, Python, Perl)

"C is a Language
That won't hold your Hand"

"C is like driving a
Formula 1 without breaks"

C Lacks…

- Exceptions
- Garbage collection
- OOP
- Polymorphism
- But…

C Lacks…

- Exceptions
- Garbage collection
- OOP
- Polymorphism
- But… it is usually faster!

How much faster?

```
cs231a@aurora ~/handout/C $ ls -1 | grep -i queens
nqueens
nqueens.c
NQueens.java
nqueens.py
```

```
cs231a@aurora ~/handout/C $ history | grep time
  541  time for N in {10..25} ; do python nqueens.py $N; done
  542  time for N in {10..25} ; do  ./nqueens $N; done
  544  time for N in {10..25} ; do  java NQueens $N -q ; done
```

```
cs231a@aurora ~/handout/C $ gcc -o nqueens nqueens.c
cs231a@aurora ~/handout/C $ gcc -O3 -o nqueens nqueens.c
```

http://www.science.smith.edu/dftwiki/index.php/N-Queens_Problem_in_Java

http://www.science.smith.edu/dftwiki/index.php/CSC231:_N-Queens_Problem:_interpreted_vs_compiled

Python

Python

Java

Python

Java

C/C++

Good Reference

- Essential C, by Nick Parlante, Stanford U.
  http://cslibrary.stanford.edu/101/EssentialC.pdf

Hello World!

```
#include <stdio.h>

int main() {
  printf("\nHello World\n");
  return 0;
}
```

- Library
- Strings
- Block-structured language
- main()

Hello World!

```
#include <stdio.h>

int main() {
   printf("\nHello World\n");
   return 0;
}
```

- Library
- Strings
- Block-structured language
- main()

getcopy C/hello.c

Compiling on Aurora

- gcc: Gnu C compiler

```
[~/handout/C]$ gcc hello.c
[~/handout/C]$ ./a.out

Hello World
```

```
[~/handout/C]$ gcc -o hello hello.c
[~/handout/C]$ ./hello

Hello World
```

Program Files

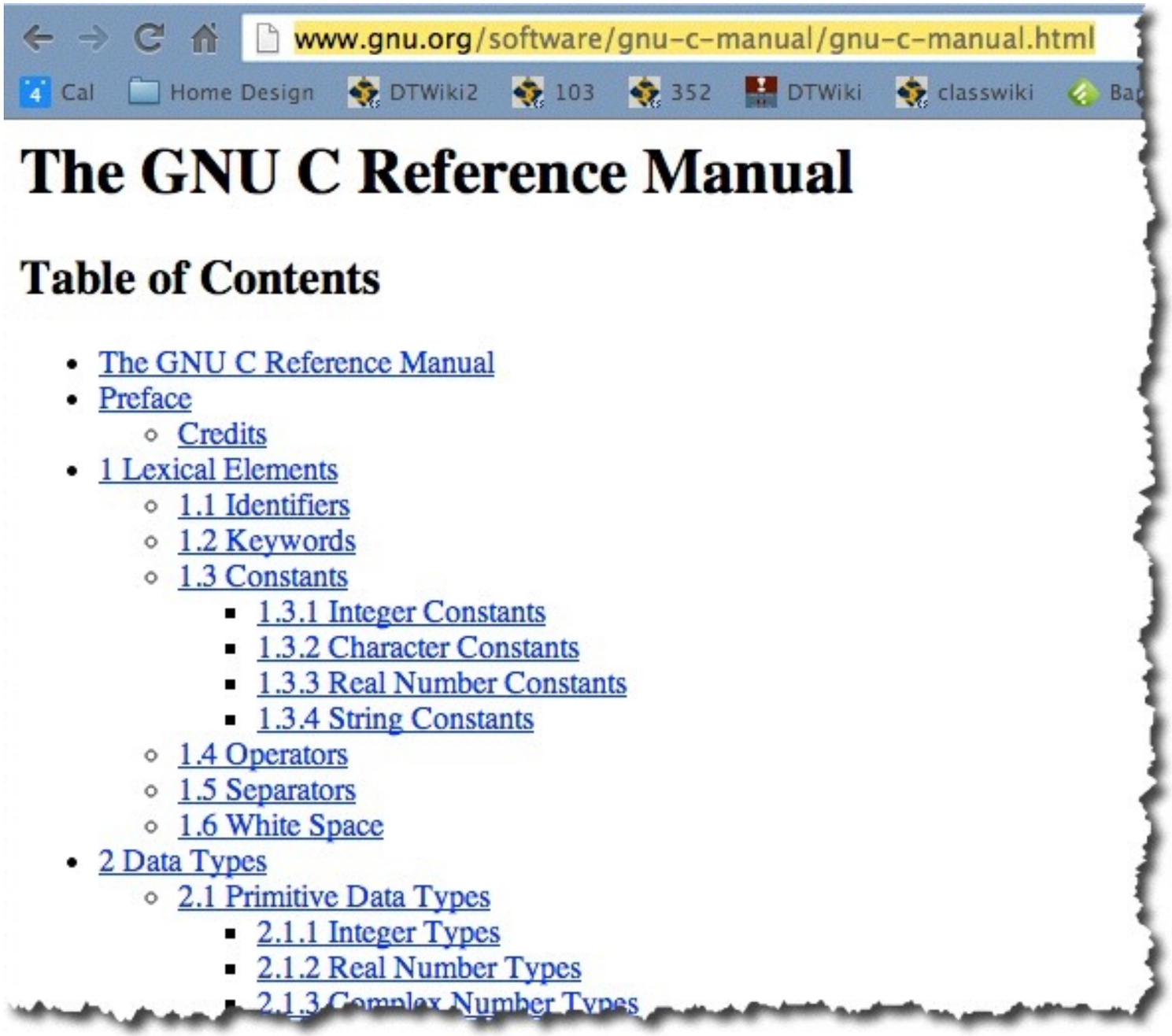
```
[~/handout/C]$ ls -l
total 28
-rwx------ 1 352a 352a 6583 Oct  6 16:41 a.out*
-rwx------ 1 352a 352a 6583 Oct  6 16:48 hello*
-rw------- 1 352a 352a   66 Oct  6 16:41 hello.c
-rw------- 1 352a 352a   67 Oct  6 16:39 hello.c~
```

Exercise

- Write your own "Hello World!" program
- Make it print something like:

```
**************
* C Rocks! *
**************
```

Good Reference on the *C Compiler*



- `man gcc`

- http://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html

Printing

- ***printf***(*"string with %-operators"*, list of vars);
  - `%d int`
  - `%f float`
  - `%s string`

We'll see examples soon!

Variables

- Simple types
- No strings!
- No booleans (only 0 for *false* and !0 for *true*)
- No classes, no objects!

```
int    -> integer variable
short  -> short integer
long   -> long integer
float  -> single precision real (floating point) variable
double -> double precision real (floating point) variable
char   -> character variable (single byte)
```

Comments

```
/*
programName.c
author

This is the header
*/
#include <stdio.h>
#include <string.h>

int main() {
    // line comment
    print( “Hello!\n" );
    return 0;
}
```

Declaring Ints and Printing Them

```
// printIntsFloat.c
#include <stdio.h>

int main() {
  int amount = 1234;
  int no20s, no10s, no5s, no1s;

  printf( "Withdrawal:  $%d\n", amount );

  no20s  = amount / 20;
  amount = amount % 20;
  no10s  = amount / 10;
  amount = amount % 10;
  no5s   = amount / 5;
  no1s   = amount % 5;

  printf( "%d $20-bill(s)\n", no20s );
  printf( "%d $10-bill(s)\n", no10s );
  printf( "%d $5-bill(s)\n",  no5s );
  printf( "%d $1-bill(s)\n",  no1s );

  return 0;
}
```

Declaring Floats and Printing Them

```
// printIntsFloat2.c

#include <stdio.h>

int main() {
  float pi = 3.14159265359;

  printf( "pi = %f|\n", pi );
  printf( "pi = %10.2f|\n", pi );
  printf( "pi = %-10.2f|\n", pi );
  printf( "pi = %20.10f|\n", pi );

  return 0;
}
```

Declaring Floats and Printing Them

```
// printIntsFloat2.c

#include <stdio.h>

int main() {
  float pi = 3.14159265359;

  printf( "pi = %f|\n", pi );
  printf( "pi = %10.2f|\n", pi );
  printf( "pi = %-10.2f|\n", pi );
  printf( "pi = %20.10f|\n", pi );

  return 0;
}
```

```
pi = 3.141593|
pi =       3.14|
pi = 3.14      |
pi =         3.1415927410|
```

Strings...

Strings

```c
#include <stdio.h>
#include <string.h>

int main() {
  char hello[] = "hello";
  char world[] = "world!";
  char sentence[100] = "";

  strcpy( sentence, hello );  // sentence <- "hello"
  strcat( sentence, " " );    // sentence <- "hello "
  strcat( sentence, world );  // sentence <- "hello world!"

  printf( "sentence = %s\n", sentence );
  return 0;
}
```

```
[~/handout/C]$ gcc stringExample.c
[~/handout/C]$ a.out

sentence = hello world!
[~/handout/C]
```

String.h

- strcat(dest, src) // appends src to dest
- strcpy(dest, src) // copies src to dest
- strlen(src) // returns length of string

C Strings end with '\0'

```
#include <stdio.h>
#include <string.h>

int main() {
  char sentence[100] = "Hello world!";

  printf( "sentence = %s\n", sentence );
  sentence[5] = '\0';
  printf( "sentence = %s\n", sentence );
  return 0;
}
```

```
~/handout]$ a.out
sentence = Hello world!
sentence = Hello
[~/handout]$
```

```
#include <stdio.h>
#include <string.h>

int main() {
   int a   = 3;
   int b   = 5;
   int c   = 0;
   char firstName[] = "your first name here";
   char lastName[] = "your last name here";
   char fullName[100];
   ...
}
```

Exercise

- make the program store a + b into c, and copy the first and last name into fullName. Make it print the information as follows:

```
~/handout]$ a.out
3 + 5 = 8
Mickey Mouse contains 12 characters.
```

Exercise

```
#include <stdio.h>
#include <string.h>

int main() {
   int a   = 3;
   int b   = 5;
   int c   = 0;
   char name[] = "your name here";
   char dash[] = "-----------------------------";
   char fullName[100];
   ...
}
```

- Enter your full name in the program and make it print your name with a line of dash below. Same # of chars in both lines.

```
~/handout/C $ a.out
Mickey Mouse
------------
```


–Johnny Appleseed
We stopped here last time…

For-Loops

```
#include <stdio.h>

int main() {
   int i;
   int sum = 0;

   // compute the sum of all the numbers from 1 to 100
   for ( i=1; i<=100; i++ ) {
      sum += i;
   }

   printf( "\nsum = %d\n\n", sum );
   return 0;
}
```

```
getcopy C/for1.c
```

For-Loops

No “int” declaration!!!

```
#include <stdio.h>

int main() {
   int i;
   int sum = 0;

   // compute the sum of all the numbers from 1 to 100
   for ( i=1; i<=100; i++ ) {
      sum += i;
   }

   printf( "\nsum = %d\n\n", sum );
   return 0;
}
```

```
getcopy C/for1.c
```

Weird For-Loop?

```
#include <stdio.h>

void main() {
  int i;
  int sum = 0;

  // loop… some number of times…  Infinitely?
  for ( i=10; i; i-- ) {
    printf( "%d\n", i );
  }

}
```

```
getcopy C/for2.c
```

While-Loops

```
#include <stdio.h>

int main() {
   int i;
   int sum = 0;

   // compute the sum of all the numbers from 1 to 100
   i = 1;
   while ( i<=100 ) {
      sum += i;     // could have also used i++
      i += 1;
   }

   printf( "\nsum = %d\n\n", sum );
   return 0;
}
```

```
getcopy C/while1.c
```

Weird While-Loop?

```
#include <stdio.h>

int main() {
   int i;
   int sum = 0;

   // compute the sum of all the numbers from 1 to 100
   i = 100;
   while ( i ) {
      sum += i--;
   }

   printf( "\nsum = %d\n\n", sum );

   return 0;
}
```

getcopy C/while3.c

Infinite Loops

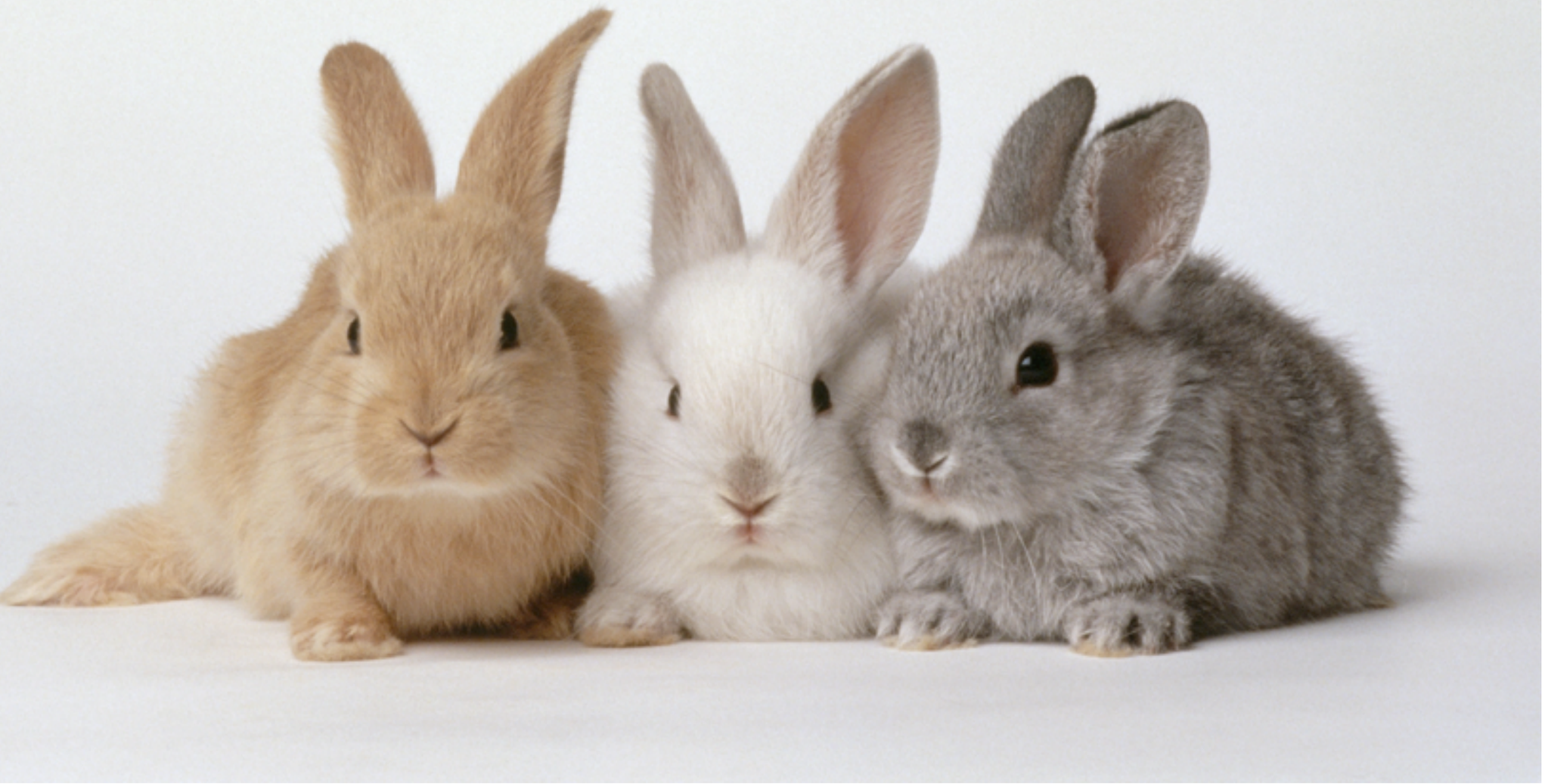
```
#include <stdio.h>

int main() {

  while ( 1 ) {
      printf( “hello!\n” );
  }
  return 0;
}
```

```
#include <stdio.h>

int main() {

  for ( ;; ) {
    printf( “hello!\n” );
  }
  return 0;
}
```

Complete These Loops

```
#include <stdio.h>
// compute the sum of 1 to 100
int main() {
  int x=100, sum=0;
  while ( 1 ) {



  }
  printf( "sum = %d\n", sum ); return 0;
}
```

```
#include <stdio.h>
// compute the sum of 1 to 100
int main() {
  int x=100, sum=0;
  for ( ;; ) {



  }
  printf( "sum = %d\n", sum ); return 0;
}
```



Exercise

```
Compute and display
the first 20 fibonacci terms
(1, 1, 2, 3…) without using an array…
```

Exercise

- Write a program that displays a triangle of *N* lines of stars:

```
*
**
***
****
*****
```

Symbolic Constants

```
#include <stdio.h>

#define NAME       "Mickey"
#define HEIGHT     5
#define YEARBORN   1928


void main() {
  printf( "%s is %d inches high, and was created in %d\n\n",
          NAME, HEIGHT, YEARBORN );
}
```

Symbolic Constants

```
#include <stdio.h>

#define NAME       "Mickey"
#define HEIGHT     5
#define YEARBORN   1928

void main() {
  printf( "%s is %d inches high, and was created in %d\n\n",
          "Mickey", 5, 1928 );
}
```

After preprocessing

Symbolic Constants ("Macros")

```
#include <stdio.h>

#define square(x) ((x) * (x))

int main() {
  int a = 1;
  int b = square( a );
  int c = square( a+b );
  int d = square( a+b-3 );

  printf( "%d %d %d %d %d\n", a, b, c, d );

  return 0;
}
```

Symbolic Constants ("Macros")

```
~/handout/C $ gcc -E macros1.c
```

```
int main() {
  int a = 1;
  int b = ((a) * (a));
  int c = ((a+b) * (a+b));
  int d = ((a+b-3) * (a+b-3));

  printf( "%d %d %d %d %d\n", a, b, c, d );

  return 0;
}
```

getcopy C/macros1.c

Conditionals

```
#include <stdio.h>

void main() {
   int a = 5;
   int b = 3;
   int c = 7;

   if ( a <= b && a <= c )
     printf( "%d is the smallest\n\n", a );
   else if ( b <= a && b <= c )
     printf( "%d is the smallest\n\n", b );
   else
     printf( "%d is the smallest\n\n", c );
}
```

Conditionals

&&	and
||	or
!	not

Conditionals (cont'd)

```
switch ( ordinal_expression ) {
  case ordinal_value: {
    // ...
    break;
  }
  case ordinal_value: {
    // ...
    break;
  }
  default: {
    // ...
  }
}
```

Conditionals (cont'd)

ints or chars, something countable

```
switch ( ordinal_expression ) {
  case ordinal_value: {
    // ...
    break;
  }
  case ordinal_value: {
    // ...
    break;
  }
  default: {
    // ...
  }
}
```

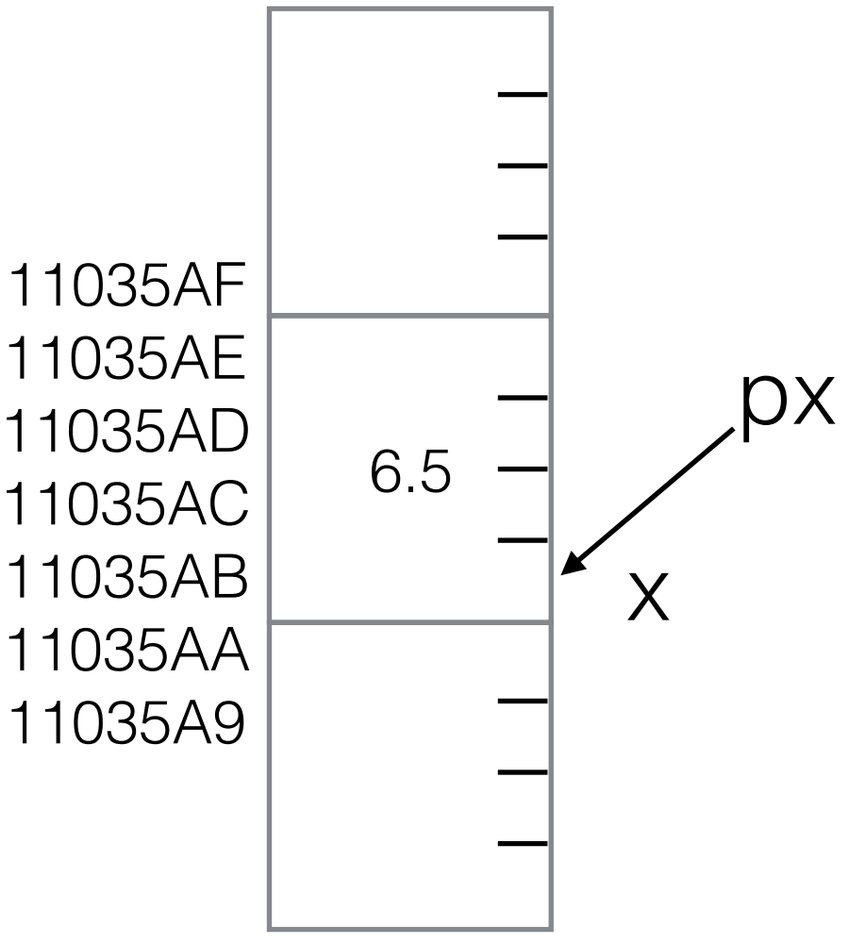
Switch/Case Example

```
#include <stdio.h>

int main () {

   /* local variable definition */
   char grade = 'B';

   switch( grade ) {
      case 'A' :
         printf("Excellent!\n" );
         break;
      case 'B' :
      case 'C' :
         printf("Well done\n" );
         break;
      case 'D' :
         printf("You passed\n" );
         break;
      case 'F' :
         printf("Better try again\n" );
         break;
      default :
         printf("Invalid grade\n" );
   }

   printf("Your grade is  %c\n", grade );

   return 0;
}
```

https://www.tutorialspoint.com/cprogramming/switch_statement_in_c.htm

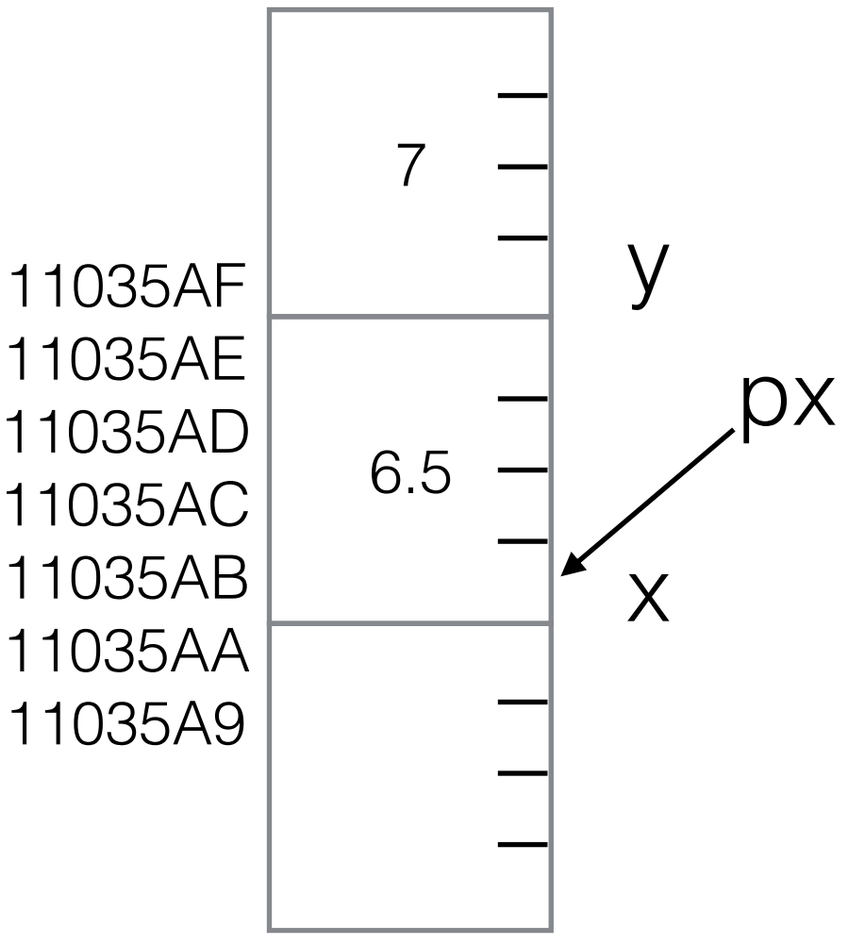
Pointers

Concept




```
float x = 6.5;
float* px = &x;
```

Concept, cont'd




```
float x = 6.5, y = 7;
float* p = &x;

*p = 10;
p = &y;
*p = 20;
```

Example: Initialize an Array

```
#include <stdio.h>
#define SIZE 10

int  main() {
     float A[SIZE];
     int i;

     for ( i=0; i<SIZE; i++ )
          A[i] = i;

    for ( i=0; i<SIZE; i++ )
          printf( "A[%d] = %1.2f\n", i, A[i]);
}
```

Using indexing

getcopy C/initArray0.c

Example: Initialize an Array

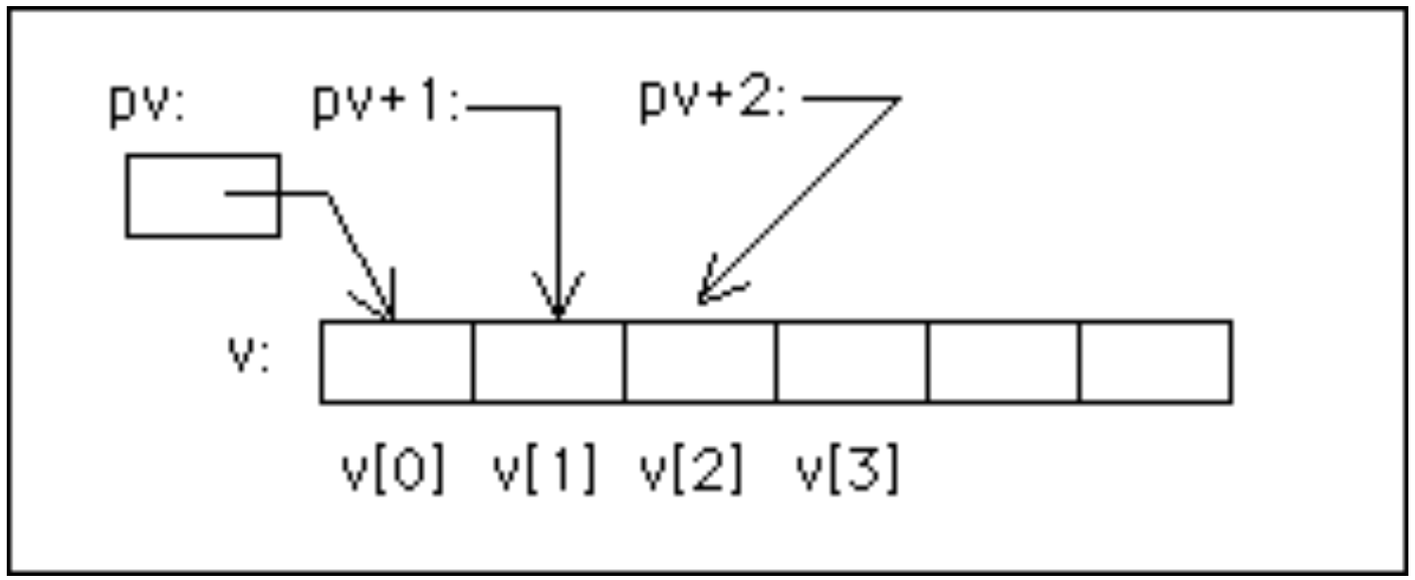
```
#include <stdio.h>
#define SIZE 10

void main() {
    float A[SIZE];
    float* p;
    int i;

    p = A;
    for ( i=0; i<SIZE; i++ ) {
        *p = i;
        p++;
    }

    p = A;
    for ( i=0; i<SIZE; i++ ) {
        printf( "p=%p A[%d] = %1.2f *p = %1.2f\n",
                 p, i, A[i], *p );
        p = p + 1;
    }
}
```

Using pointers

getcopy C/initArray.c

```
./a.out

p=0x7fff88d54560 A[0] = 0.00 *p = 0.00
p=0x7fff88d54564 A[1] = 1.00 *p = 1.00
p=0x7fff88d54568 A[2] = 2.00 *p = 2.00
p=0x7fff88d5456c A[3] = 3.00 *p = 3.00
p=0x7fff88d54570 A[4] = 4.00 *p = 4.00
p=0x7fff88d54574 A[5] = 5.00 *p = 5.00
p=0x7fff88d54578 A[6] = 6.00 *p = 6.00
p=0x7fff88d5457c A[7] = 7.00 *p = 7.00
p=0x7fff88d54580 A[8] = 8.00 *p = 8.00
p=0x7fff88d54584 A[9] = 9.00 *p = 9.00
```

Arrays and Pointers




```
TYPE v[DIM];
TYPE* pv;

pv = v;
```

Arrays and Pointers

- The name of an array is a pointer to the first cell of the array.

```
char name[DIM];
```

- **`name`** is the same as `&(name[0])`

* and & operators

- * has two meanings, depending on context
  - “Pointer to”
  - “Contents of”
- & means “the address of”

* and & Operators

```
int A[DIM];
int* p = A;              // “int pointer p”
int *q = &A[0];          // “int pointer q”


*p = 3;                  // what p is pointing to gets 3
*(q+1) = 5;              // what q+1 is pointing to gets 5
```

We stopped here last time…

Exercise

```
#define DIM 10
int A[DIM];
int B[DIM];
int i;

init(A); // will initialize A

for ( i=0; i<DIM; i++ ) {
    B[i] = A[i];
}
```

- The program above copies Array A into Array B using indexing. Rewrite the loop so that A is copied into B using pointers.

```
File Edit Options Buffers Tools C Help
#include <stdio.h>
#define DIM 10
int A[DIM];
int B[DIM];
int i;
int *p, *q;

void main() {
  for ( i=0; i<DIM; i++ )
    A[i] = 13*i % 11;


  p = A;
  q = B;
  //for ( i=0; i<DIM; i++ ) {
  //  *(q+i) = *(p+i);
  //}

  //for ( i=0; i<DIM; i++ ) {
  //  *q = *p;
  //  q++;
  //  p++;
  //}
  for ( i=0; i<DIM; i++ )
    *(q++) = *(p++);


  for ( i=0; i<DIM; i++ )
    printf( "%d %d\n", A[i], B[i] );
}

-UU-:----F1  copyAintoB.c   All L1      (C/l Abbrev) ----------------------------
Loading cc-langs...done
```

getcopy C/copyAintoB.c

Exercise

```
#define DIM 10
int A[DIM];
int i;

for ( i=0; i<DIM; i++ )
   A[i] = 13*i % 11;
```

- Complete the code above so that the program will find the largest integer in Array A, using pointers.

```
File Edit Options Buffers Tools C Help
// findLargest.c
// D. Thiebaut
// Finds the largest element of an array of ints
#include <stdio.h>
#include <limits.h>   // to use INT_MIN
#define DIM 10


int main() {
  int A[DIM];
  int i;

  //--- initialize A ---
  for ( i=0; i<DIM; i++ )
    A[i] = 13*i % 11;

  //--- find largest element of A ---
  int max = INT_MIN;
  int *p = A;
  for ( i=0; i<DIM; i++ ) {
    printf( "A[%d] = %d\n", i, *p );

    if ( *p > max ) max = *p;
    p++;
  }
  printf( "Max = %d\n", max );

-UU-:**--F1  findLargest.c   Top L14     (C/
```

getcopy C/findLargest.c

Exercise

Duffy Duck, (413) 585-2700, xxxxxxxx
Mickey Mouse, (617) 123-4567, yyyyyyyyy
Minnie Mouse, (617) 123-4567, zzzzzz zzz
Bruno The Dog, (212) 678-9999, woof woof

- Given a list of names and personal information, blank out the phone numbers, leaving only the area code visible.

- Hints: **strstr()** is a function that will search for a string inside another string and return a pointer to the occurrence of the string, if it's there, or 0.
  str(haystack, needle)

See http://www.cplusplus.com/reference/cstring/strstr/

strstr

strstr will treat the strings as immutable

Portability

In C, this function is only declared as:

```
char * strstr ( const char *, const char * );
```

instead of the two overloaded versions provided in C++.

Example

```
/* strstr example */
#include <stdio.h>
#include <string.h>

int main ()
{
  char str[] ="This is a simple string";
  char * pch;
  pch = strstr (str,"simple");
  strncpy (pch,"sample",6);
  puts (str);
  return 0;
}
```

```
File Edit Options Buffers Tools C Help
// blanOutPhone.c
// D. Thiebaut
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void main() {
  char line[] = "Mickey Mouse, (617) 123-4567, alalalalalalalal";
  char *p, *q;
  int i;

  printf( "%s\n", line );
  p = strstr( line, ")" );
  if ( p!=NULL ) {
    //printf( "found it\n" );
    p = p+2;
    for (i=0; i<8; i++ ) {
      if ( *p != '-' )
        *p = 'x';
      p++;
    }
  }

  printf( "%s\n", line );
}
```

```
getcopy C/blankOutPhone.c
```

```
-UU-:**--F1  blankOutPhone.c   All L2     (C/l Abbrev) ----------
Auto-saving...done
```

Functions in C

Functions

- Functions are always declared **before** they are used
- Functions can return values of simple types (int, char, floats), and also *pointers*
- Functions get parameters of simple types, as well as *pointers*
- Passing by value is **automatic**. Passing by reference requires **passing a pointer to the var**

Example 1

```
#include <stdio.h>

int sum( int a, int b ) {
    return a+b;
}

int main() {
    int x = 10;
    int y = 20;
    int z;

    z = sum( x, y );
    printf( "z = %d\n", z );

    z = sum( 3, 8 );
    printf( "z = %d\n", z );

    printf( "sum( 11, 22) = %d\n", sum( 11, 22 ) );
    return 0;
}
```

```
z = 30
z = 11
sum( 11, 22) = 33
```

Example/Exercise

(Incomplete code… Add missing elements!)

```
#include <stdio.h>

void sum2( int a, int b, int  c ) {
    c = a+b;
}

int main() {
    int x = 10;
    int y = 20;
    int z;

    sum2( x, y,   z );
    printf( "z = %d\n", z );

    sum2( 3, 8,   x );
    printf( "x = %d\n", x );
    return 0;
}
```

Pass by Reference!

```
z = 30
x = 11
```

```
File Edit Options Buffers Tools C Help
// PassByReferenceSum2
// D. Thiebaut

#include <stdio.h>

void sum2( int a, int b, int *c ) {
  *c = a+b;
}

void main() {
  int x = 10;
  int y = 20;
  int z;

  sum2( x, y, &z );
  printf( "z = %d\n", z );

  sum2( 3, 8, &x );
  printf( "x = %d\n", x );
}
```

```
getcopy C/passByReferenceSum2.c
```

```
-UU-:----F1  passByReferenceSum2.c   All L1      (C/l Abbrev) --------
Loading cc-langs...done
```

Input: pass by reference!

```
#include <stdio.h>

int main() {
    int age;
    float myPi;
    char name[80];

    printf( "Enter your name, please: " );
    fgets( name, sizeof(name), stdin );
                    // will truncate to first
                    // 80 chars entered
    printf( "Enter your age: " );
    scanf( "%d", &age );

    printf( "Enter your version of pi: " );
    scanf( "%f", &myPi );

    printf( "%s is %d years old, and thinks pi is %1.10f\n\n",
            name, age, myPi );
    return 0;
}
```

Input (cont'd)

```
a.out
Enter your name, please: Mickey
Enter your age: 21
Enter your version of pi: 3.14159
Mickey is 21 years old, and thinks pi is 3.1415901184
```

```
#include <stdio.h>
#include <stdlib.h>

#define   N 10

// functions go here...

int main() {
    int A[N] = { 3, 2, 1, 0, 6, 5, 9,  8, 7, -3 };

    // your code goes here
}
```

- Write another program that finds the largest and smallest ints in A using two functions, min(), max(). The results are passed back using a **return** statement.

```
File Edit Options Buffers Tools C Help
// findMinMaxSum.c

#include <stdio.h>
#include <stdlib.h>

#define N 10

int min( int* A ) {
  int i, m = A[0];
  for ( i=1; i<N; i++ )
    if ( A[i] < m )  m = A[i];
  return m;
}

int max( int* A ) {
  int i, m = A[0];
  for ( i=1; i<N; i++ )
    if ( A[i] > m )  m = A[i];
  return m;
}

int sum( int A[] ) {
  int i, s=0;
  for ( i=0; i<N; i++ )
    s += A[i];
  return s;
}

int main( int argc, char *argv[] ) {
  int A[N] = {1, 3, 2, 4, 5, 7, 6, 9, 10, 8};

  printf( "min = %d max = %d sum = %d\n", min(A),
          max(A), sum(A) );
  return 0;
}

-UU-:**--F1  findMinMaxSum.c   All L15    (C/l Abbrev) ----------------------
```

getcopy C/findMinMaxSum.c

Exercise

- Modify the **min()** function in your previous program so that it returns the sum of all the elements in the array in a variable passed by reference.

  Example:

```
int s;
sum( A,     s );     //incomplete code
printf( "%d\n", s );
```

We stopped here last time…

Side-Step: Documenting Code

```
getcopy hw7a_goodDoc.asm
```

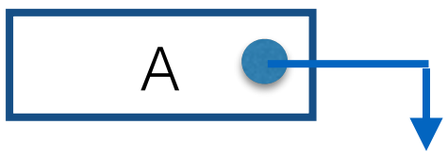
Dynamic Variables

- Dynamic: think "**new**" in Java

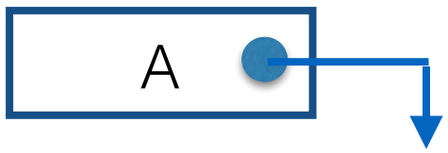
- Some variable/structure that didn't exist when the program started are added to the data area. (Think *linked lists*)

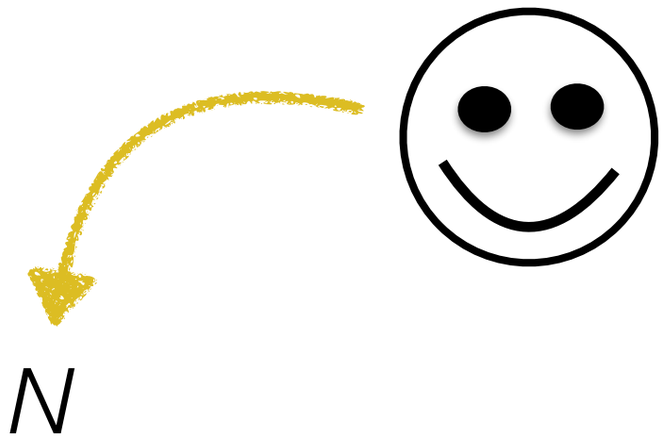
- **malloc() = M**emory **Alloc**ation for New Data Structures

Example

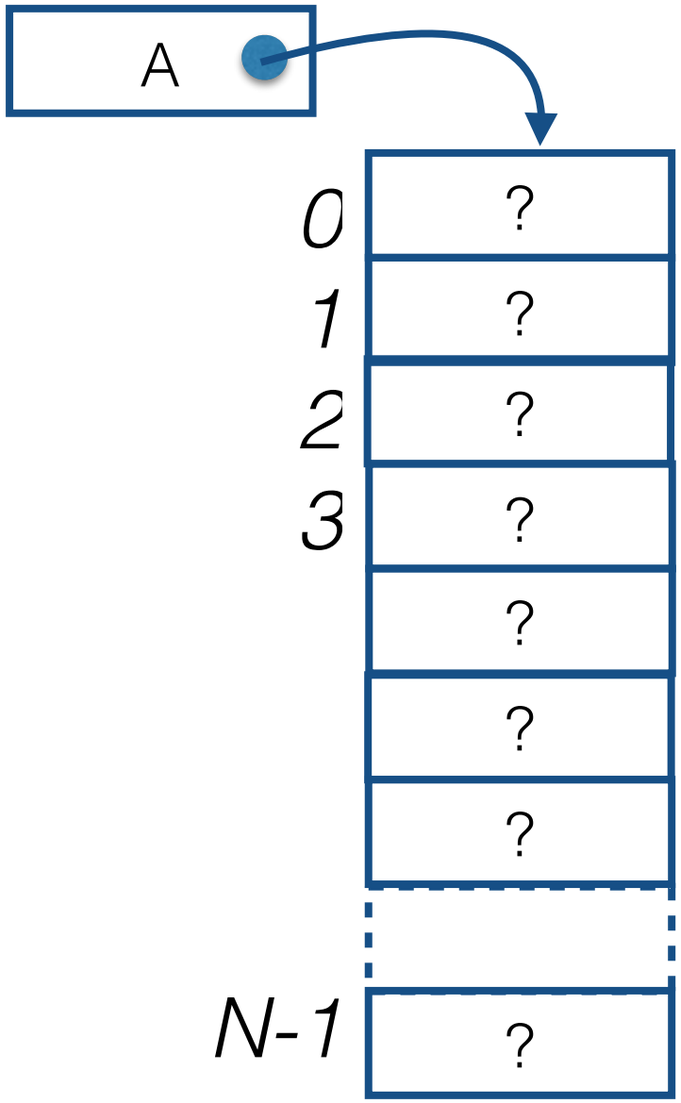


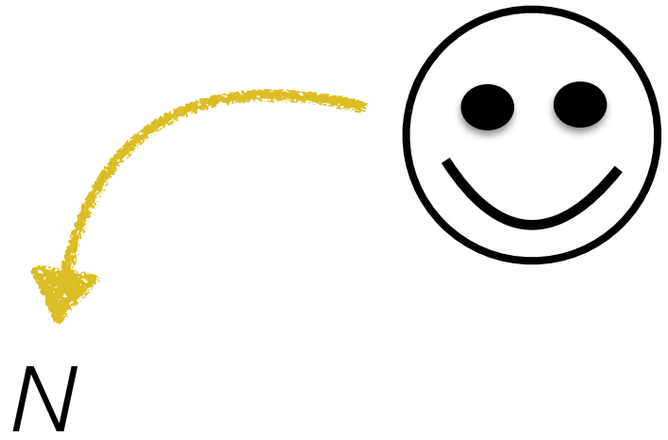
Example



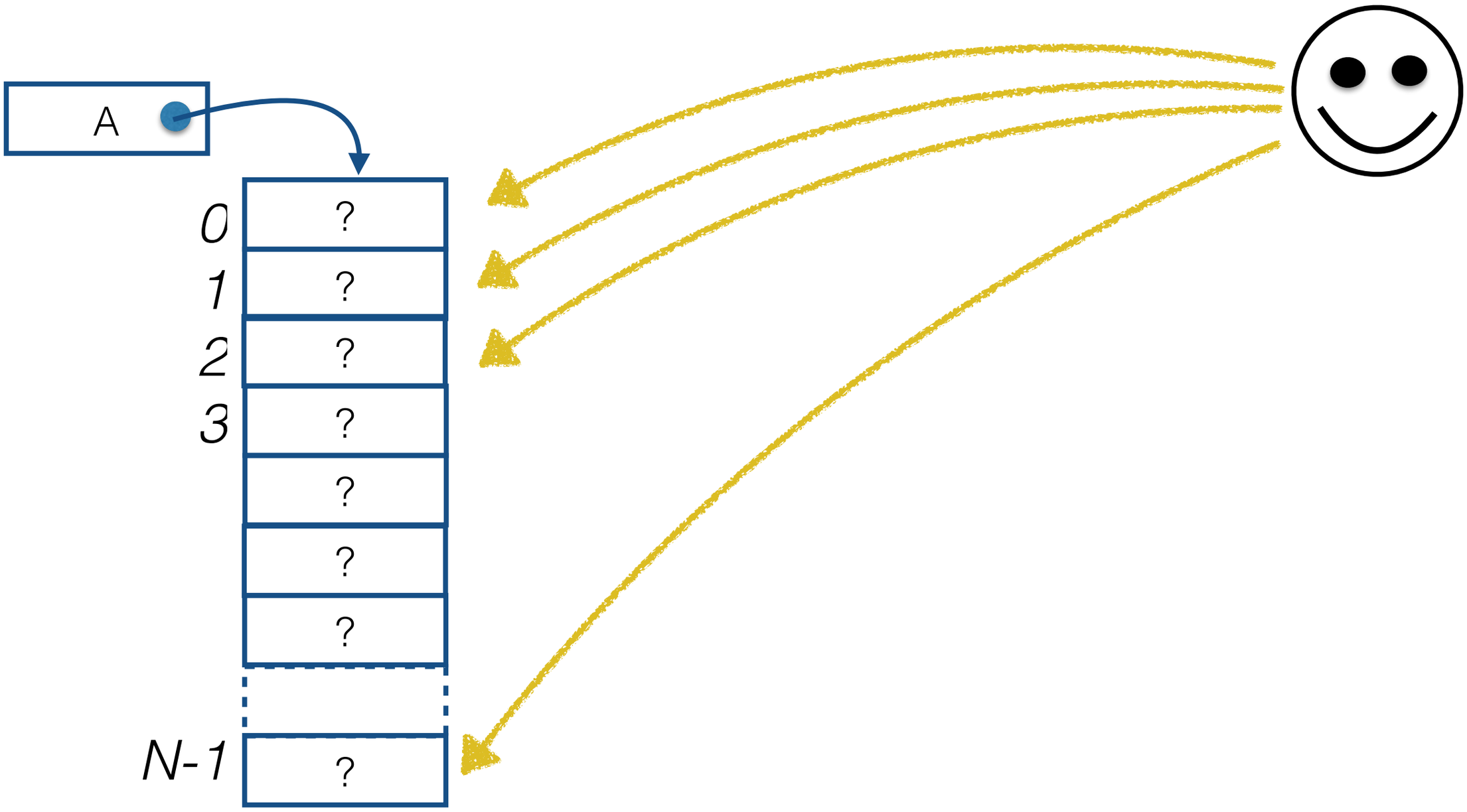


Example

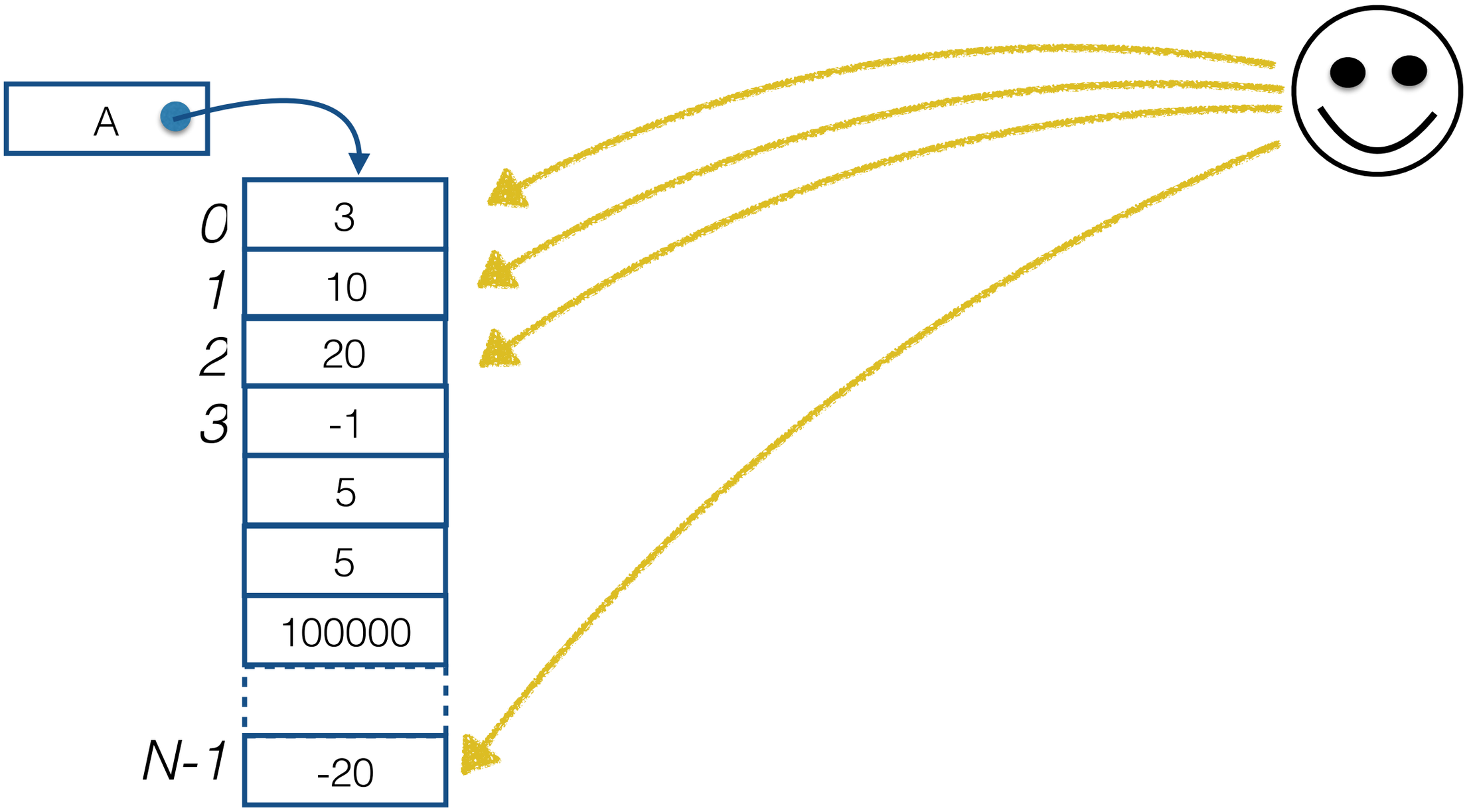




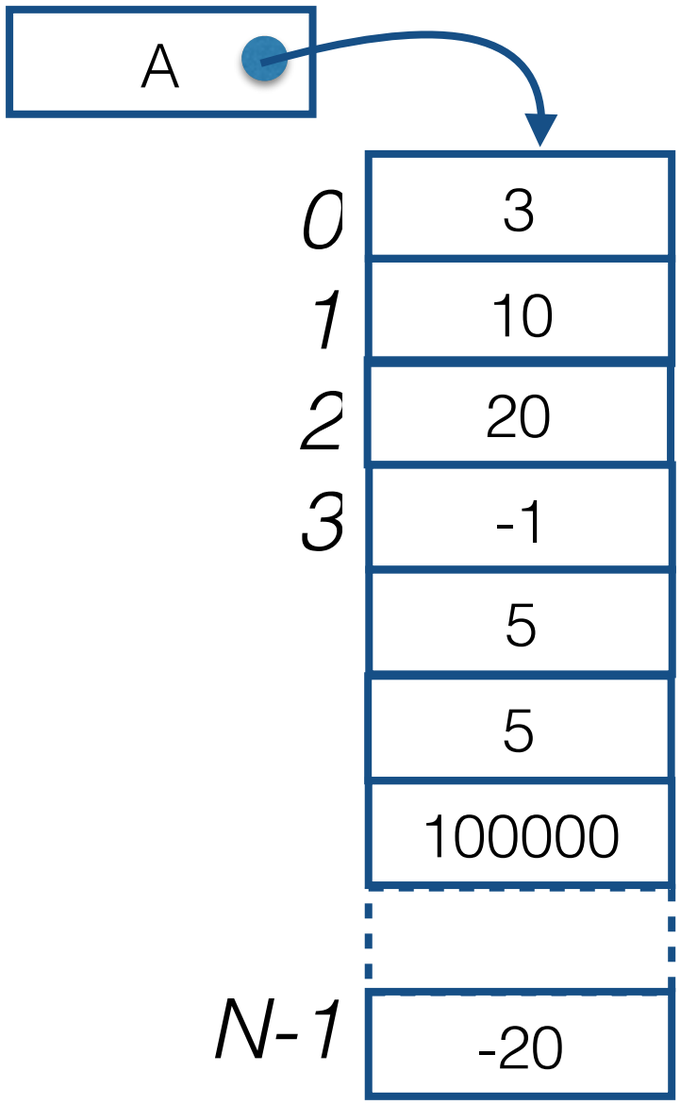
Example

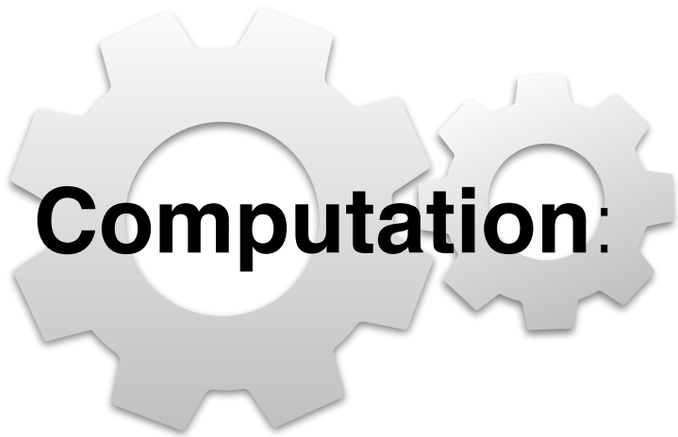


Example



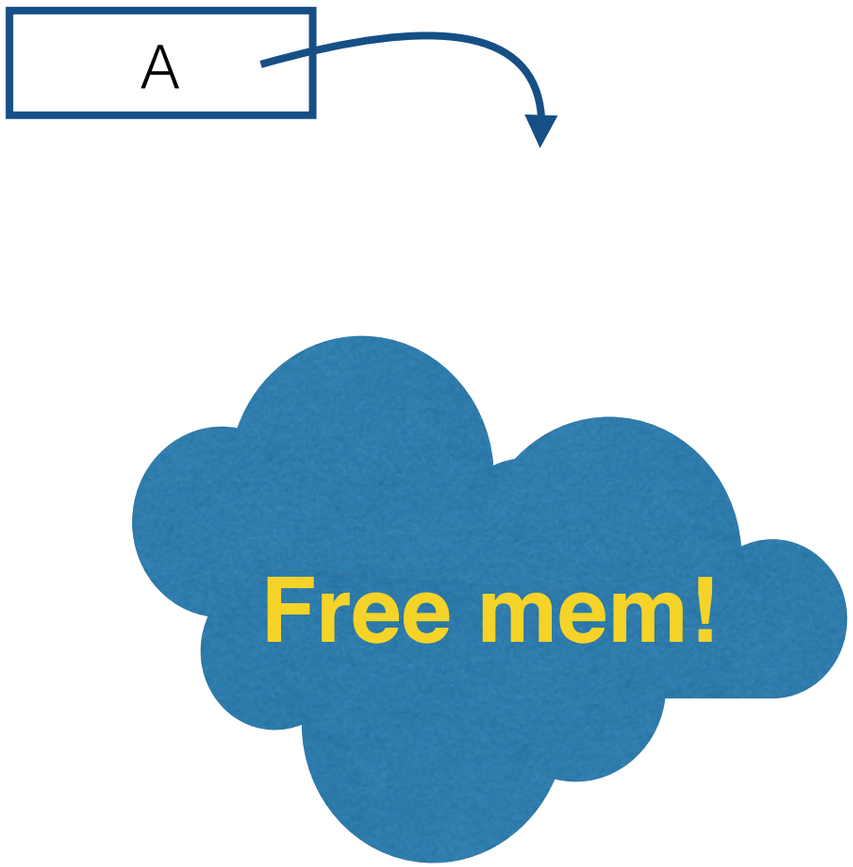
Example



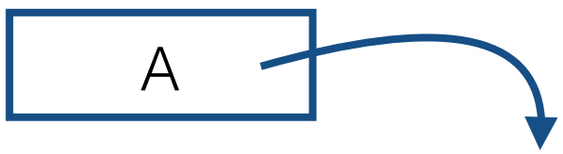


Computation: smallest = -20

Example



Example



Definition: dangling pointer!

Malloc

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int *A, N, i, smallest;

    printf( "How many ints? " );
    scanf( "%d", &N );
    A = (int *) malloc( N * sizeof( int ) );

    for ( i=0; i<N; i++ ) {
        printf( "> " );
        scanf( "%d", &(A[i]) );
    }

    smallest = A[0];
    for ( i=1; i<N; i++ )
        if ( A[i] < smallest )
            smallest = A[i];
    free( A );

    printf( "The smallest = %d\n", smallest );
    return 0;
}
```

getcopy C/malloc1.c

Exercise

- Write a program that gets 2 ints from the user, *low* and *high*, and create an array of ints containing all the integers included between *low* and *high*.
  cs231a@aurora ~/handout/C $ ./a.out

```
Enter low & high bounds
> 10
> 23
Creating array of 14 integers
10
11
12
13
14
15
16
17
18
19
20
21
22
23
cs231a@aurora ~/handout/C $
```

sizeof()
can be tricky…

http://theplaceofpraise.com/wp-content/uploads/2016/05/be-careful-300x225.png

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main( int argc, char* argv[] ) {
  int A[] = { 1, 2, 3, 4, 5 };
  int *B = (int *) malloc( 5 * sizeof( int ) );
  int *p = A;
  char name[] = "Smith College";
  int a = 3;
  float x = 3.14159;
  int i;

  for ( i=0; i<5; i++ ) B[i] = i;

  printf( "sizeof(A)    = %lu\n", sizeof( A ) );
  printf( "sizeof(A[0]) = %lu\n", sizeof( A[0] ) );
  printf( "sizeof(B)    = %lu\n", sizeof( B ) );
  printf( "sizeof(B[0]) = %lu\n", sizeof( B[0] ) );
  printf( "sizeof(p)    = %lu\n", sizeof( p ) );
  printf( "sizeof(*p)   = %lu\n", sizeof( *p ) );
  printf( "sizeof(name) = %lu\n", sizeof( name ) );
  printf( "strlen(name) = %lu\n", strlen( name ) );
  printf( "sizeof(a)    = %lu\n", sizeof( a ) );
  printf( "sizeof(x)    = %lu\n", sizeof( x ) );

  return 0;
}
```

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main( int argc, char* argv[] ) {
  int A[] = { 1, 2, 3, 4, 5 };
  int *B = (int *) malloc( 5 * sizeof( int ) );
  int *p = A;
  char name[] = "Smith College";
  int a = 3;
  float x = 3.14159;
  int i;

  for ( i=0; i<5; i++ ) B[i] = i;

  printf( "sizeof(A)    = %lu\n", sizeof( A ) );
  printf( "sizeof(A[0]) = %lu\n", sizeof( A[0] ) );
  printf( "sizeof(B)    = %lu\n", sizeof( B ) );
  printf( "sizeof(B[0]) = %lu\n", sizeof( B[0] ) );
  printf( "sizeof(p)    = %lu\n", sizeof( p ) );
  printf( "sizeof(*p)   = %lu\n", sizeof( *p ) );
  printf( "sizeof(name) = %lu\n", sizeof( name ) );
  printf( "strlen(name) = %lu\n", strlen( name ) );
  printf( "sizeof(a)    = %lu\n", sizeof( a ) );
  printf( "sizeof(x)    = %lu\n", sizeof( x ) );

  return 0;
}
```

```
sizeof(A)    = 20
sizeof(A[0]) = 4
sizeof(B)    = 8
sizeof(B[0]) = 4
sizeof(p)    = 8
sizeof(*p)   = 4
sizeof(name) = 14
strlen(name) = 13
sizeof(a)    = 4
sizeof(x)    = 4
```

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main( int argc, char* argv[] ) {
  int A[] = { 1, 2, 3, 4, 5 };
  int *B = (int *) malloc( 5 * sizeof( int ) );
  int *p = A;
  char name[] = "Smith College";
  int a = 3;
  float x = 3.14159;
  int i;

  for ( i=0; i<5; i++ ) B[i] = i;

  printf( "sizeof(A)    = %lu\n", sizeof( A ) );
  printf( "sizeof(A[0]) = %lu\n", sizeof( A[0] ) );
  printf( "sizeof(B)    = %lu\n", sizeof( B ) );
  printf( "sizeof(B[0]) = %lu\n", sizeof( B[0] ) );
  printf( "sizeof(p)    = %lu\n", sizeof( p ) );
  printf( "sizeof(*p)   = %lu\n", sizeof( *p ) );
  printf( "sizeof(name) = %lu\n", sizeof( name ) );
  printf( "strlen(name) = %lu\n", strlen( name ) );
  printf( "sizeof(a)    = %lu\n", sizeof( a ) );
  printf( "sizeof(x)    = %lu\n", sizeof( x ) );

  return 0;
}
```

Different!

```
sizeof(A)    = 20
sizeof(A[0]) = 4
sizeof(B)    = 8
sizeof(B[0]) = 4
sizeof(p)    = 8
sizeof(*p)   = 4
sizeof(name) = 14
strlen(name) = 13
sizeof(a)    = 4
sizeof(x)    = 4
```

Reading & Writing Files

File I/O output

```c
#include <stdio.h>

int main() {
    FILE *fp;
    int i;
    char name[] = "Smith College";

    fp = fopen("hello.txt", "w");  // open file for writing

    fprintf(fp, "\nHello " );      // write constant string
    fprintf(fp, "%s\n\n", name );  // write string

    fclose(fp);                    // close file
    return 0;
}
```

getcopy C/helloFile.c

File I/O: Reading Text

```c
#include <stdio.h>

int main() {
    FILE *fp;
    char line[80];

    fp = fopen( "hello.txt", "r" ); // open file for reading

    while ( !feof( fp ) ) {          // while not eof
        fgets( line, 80, fp );       // get at most 80 chars
        if ( feof( fp ) )            // if eof reached stop
            break;
        line[79] = '\0';             // truncate line to be safe
        printf( "%s", line );        // print it
    }

    fclose( fp );                    // close file
    return 0;
}
```

getcopy C/readFile.c

File I/O: Input Numbers

```
[~handout] cat data.txt
4
110
10
20
10
```

File I/O: Reading Ints

```
#include <stdio.h>
int N; // global
int main() {
    FILE *fp;
    char line[80];
    int *A, n1, i;

    fp = fopen( "data.txt", "r" ); // 1st number is # of lines
                                   // then 3 ints per line
    if ( feof( fp ) ) {
        printf( "Empty file!\n\n" );
        return 1;
    }

    //-- get the number of lines --
    fscanf( fp, "%d", &N );
    A = (int *) malloc( N * sizeof( int ) );
    i = 0;

    while ( !feof( fp ) ) {
      fscanf( fp, "%d", &(A[i] ) );
      if ( feof( fp ) || i >= N )
          break;
      i++;
    }
    //-- close the file --
    fclose( fp );
    for ( i=0; i<N; i++ ) printf( "%d\n", A[i] );
    return 0;
}
```

getcopy C/readFileNumbers0.c

Exercise

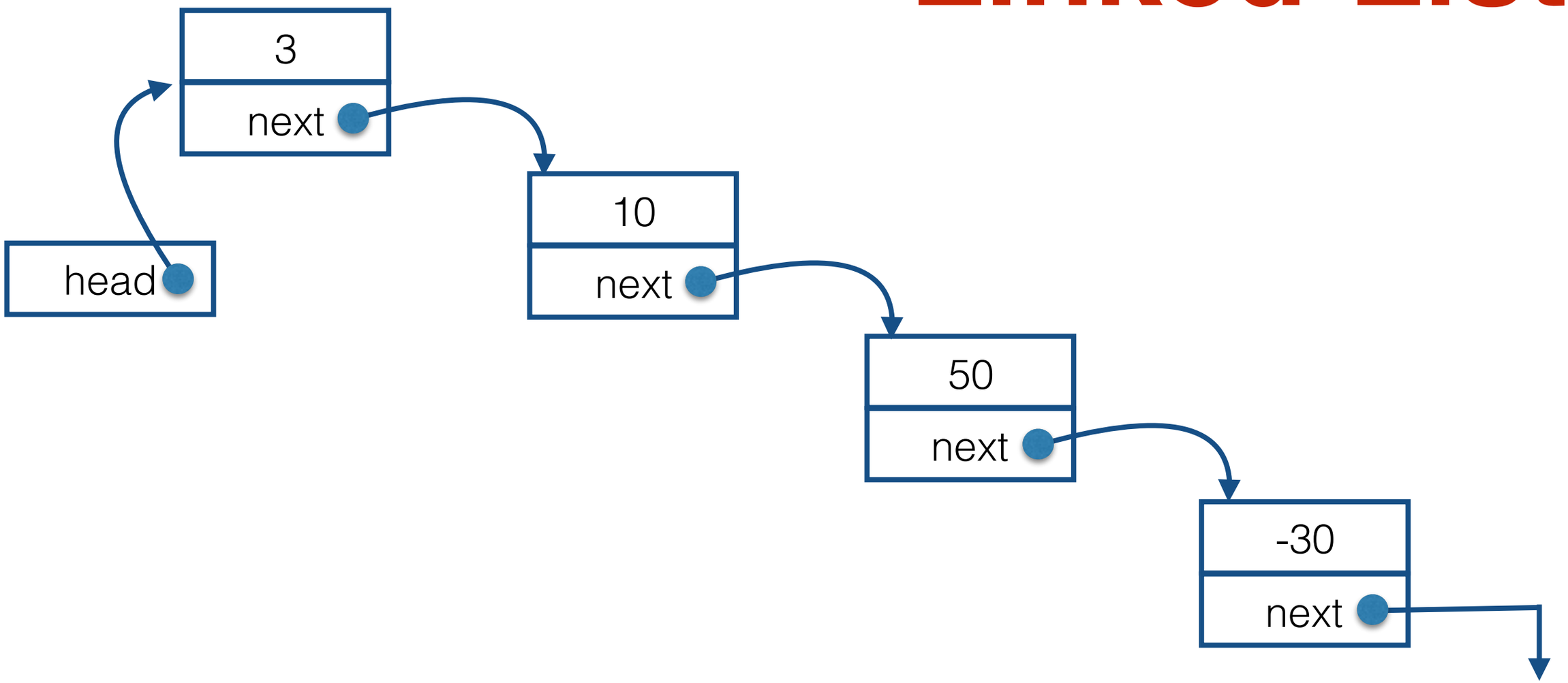
- Take the previous program and put the code reading the file in a function, so that the main program becomes:

```
File Edit Options Buffers Tools C Help

int main() {
    int *A, i;

    A = getArray( "data.txt" );

    //--- display array ---
    for ( i=0; i<N; i++ )
      printf( "%d\n", A[i] );

    return 0;
}



-UU-:----F1  readFileNumbersFunc.c   Bot L47     (C/l Abbrev) ---
```

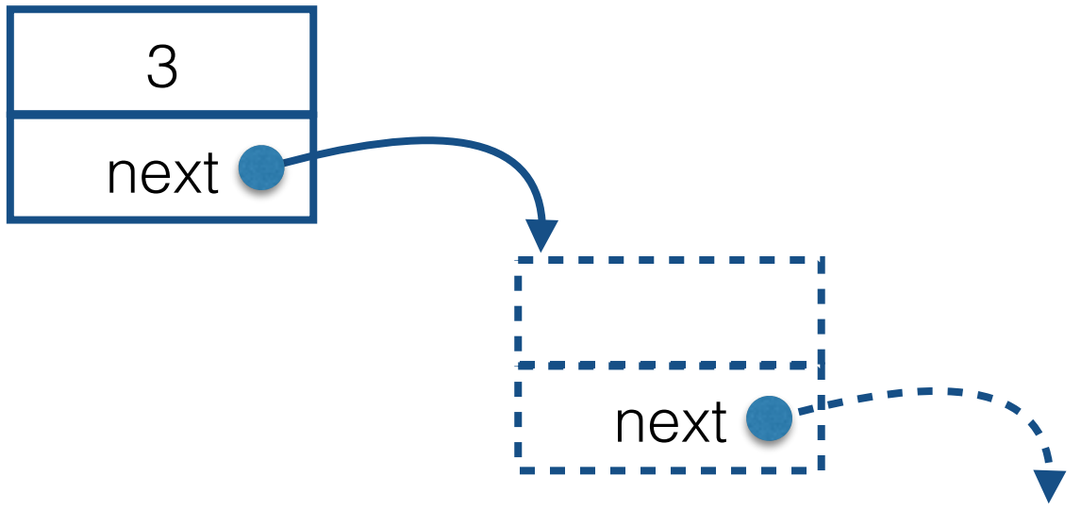
```
File Edit Options Buffers Tools C Help
int N; // dimension of array.
       // global to all functions

int* getArray( char* fileName ) {
  FILE *fp;
  int *A, i;
  fp = fopen( fileName, "r" ); // 1st number is # of lines
                               // then 3 ints per line
  if ( feof( fp ) ) {
    printf( "Empty file!\n\n" );
    return NULL;
  }

  //--- get the number of lines ---
  fscanf( fp, "%d", &N );
  A = (int *) malloc( N * sizeof( int ) );
  i = 0;

  //--- get all N numbers ---
  while ( !feof( fp ) ) {
    fscanf( fp, "%d", &(A[i] ) );
    if ( feof( fp ) || i >= N )
      break;
    i++;
  }

  fclose( fp );
  //--- return array ---
  return A;
}
-UU-:----F1  readFileNumbersFunc.c   19% L25     (C/l Abbrev) ----------
```

getcopy C/readFileNumbersFunc.c

Linked-Lists

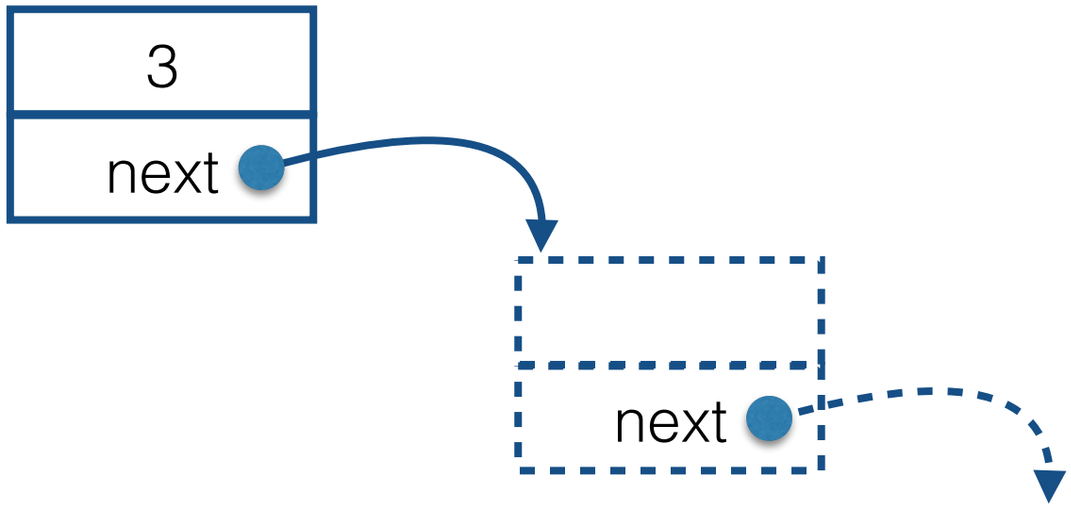


Linked-Lists



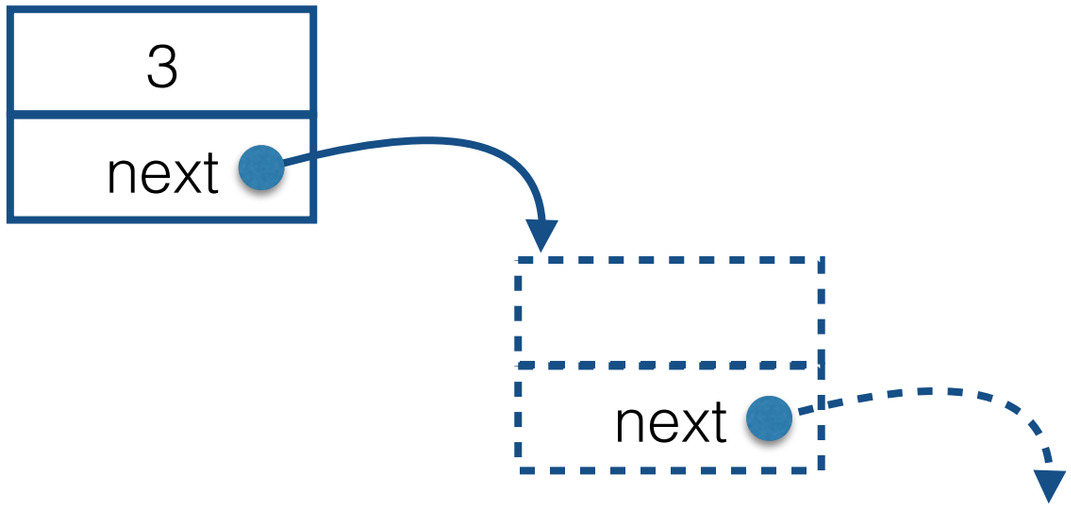

```
File Edit Options Buffers Tools C Help
#include <stdio.h>
#include <stdlib.h>

struct node {
   int item;
   struct node * next;
};

-UU-:**--F1  linkedList.c    Top L9      (C/l Abbrev) -----
```

Linked-Lists




```
File Edit Options Buffers Tools C Help
#include <stdio.h>
#include <stdlib.h>

struct node {
  int item;
  struct node * next;
};
typedef struct node node_t;

-UU-:**--F1  linkedList.c    Top L1      (C/l Abbrev) ----------
```

Linked-Lists

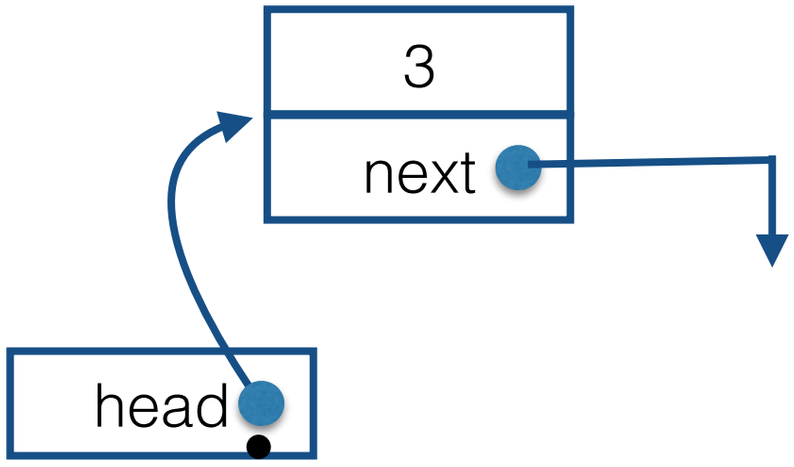


node_t

```
File Edit Options Buffers Tools C Help
#include <stdio.h>
#include <stdlib.h>

struct node {
  int item;
  struct node * next;
};
typedef struct node node_t;

-UU-:**--F1  linkedList.c   Top L1     (C/l Abbrev) ----------
```

new type

Exercise



- Create a 1-node Linked list with 3 for item.

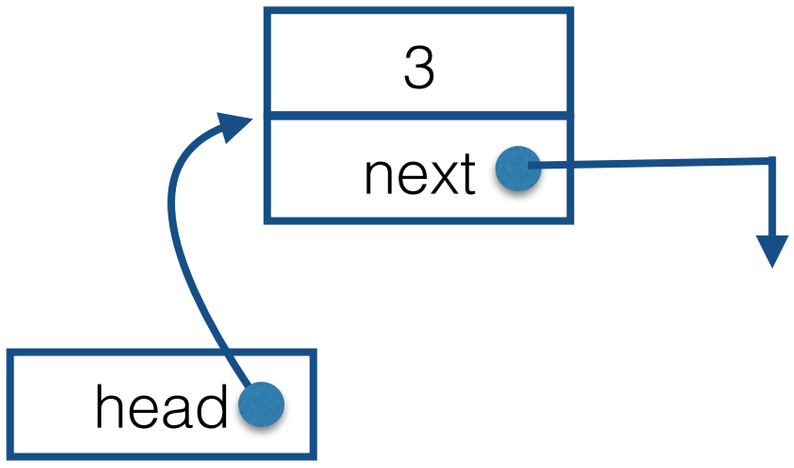
Notation

```
node_t    *p;

p = (node_t *) malloc( sizeof( node_t ) );
(*p).item = 3;
(*p).next = NULL;
```

```
node_t    *p;

p = (node_t *) malloc( sizeof( node_t ) );
p->item = 3;
p->next = NULL;
```

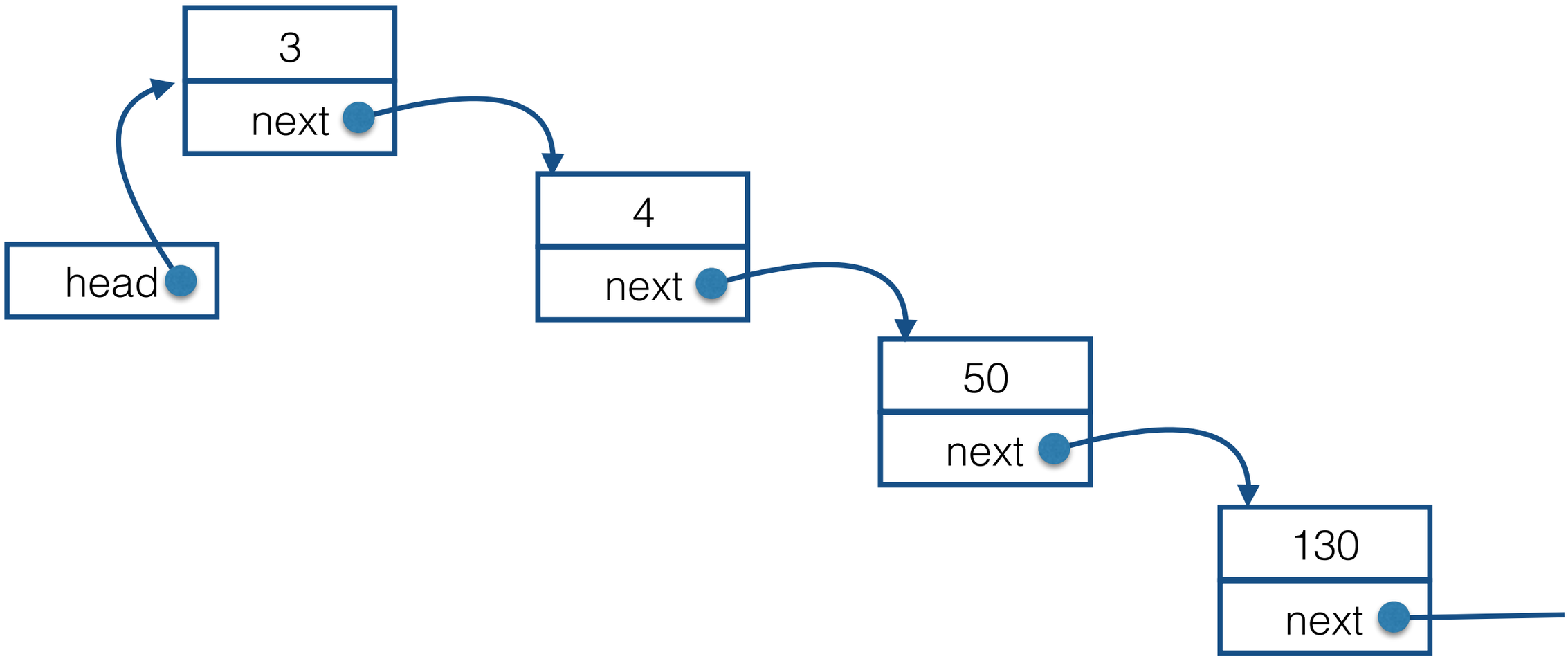
```
File Edit Options Buffers Tools C Help
*/
#include <stdio.h>
#include <stdlib.h>

struct node {
  int item;
  struct node * next;
};
typedef struct node node_t;


int main() {
  node_t *head = NULL, *p;
  int i;


  for ( i = 1; i<20; i+= 2 ) {
    p = (node_t *) malloc( sizeof( node_t ) );
    p->item = i;
    p->next = head;
    head    = p;
  }

  for ( p = head; p != NULL; p = p->next ) {
    printf( "%d\n", p->item );
  }

-UU-:**--F1  linkedList.c   10% L15    (C/l Abbrev) ----------------------------
```

Exercise



- Write the code that searches a linked list for a particular value (int)

Exercise



- Insert a new node with 10 for its item, after the node containing 4.

The End