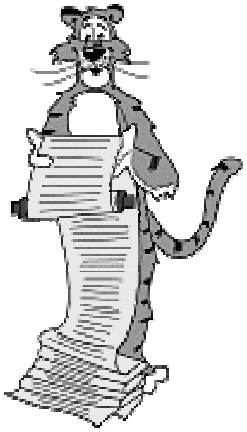
Lecture P2: Arrays



Arrays

Built-in to C.

- Declare using [ ].

```
double a0, a1, a2, a3, a4, a5, a6, a7, a8, a9;
```

vs.

```
double a[10];
```

- To access element i of array named a, use `a[i]`.
- Caveats:
  -
  -
- Limitation: need to fix size of array ahead of time.

Array Example: Manipulate Polynomials

$$p(x) = c_9 x^9 + c_8 x^8 + c_7 x^7 + c_6 x^6 + c_5 x^5 + c_4 x^4 + c_3 x^3 + c_2 x^2 + c_1 x^1 + c_0$$

C representation of $1.2x^9 + 3.8x^5 + 7.0$.

```
int i;
double c[10];
for (i = 0; i < 10; i++)
  c[i] = 0.0;

c[0] = 7.0;
c[5] = 3.8;
c[9] = 1.2;
```

initialize variables before using

Possible memory representation (assuming array starts at 107).

Memory Address	107	108	109	110	111	112	113	114	115	116
Variable	c[0]	c[1]	c[2]	c[3]	c[4]	c[5]	c[6]	c[7]	c[8]	c[9]
Value	7.0	0.0	0.0	0.0	0.0	3.8	0.0	0.0	0.0	1.2

Array Example: Manipulate Polynomials

$$p(x) = c_9 x^9 + c_8 x^8 + c_7 x^7 + c_6 x^6 + c_5 x^5 + c_4 x^4 + c_3 x^3 + c_2 x^2 + c_1 x^1 + c_0$$

Evaluating p(x) at x = 3.14.

```
double x = 3.14, p = 0.0;
int i;
for (i = 0; i < 10; i++)
   p += c[i] * pow(x, i);
```

exponentiation

Clever alternative (Horner's method).

```
double x = 3.14, p = 0.0;
int i;
for (i = 9; i >= 0; i--)
   p += c[i] + (x * p);
```

Array Example: Manipulate Polynomials

$$p(x) = c_9 x^9 + c_8 x^8 + c_7 x^7 + c_6 x^6 + c_5 x^5 + c_4 x^4 + c_3 x^3 + c_2 x^2 + c_1 x^1 + c_0$$

Differentiating.

$$p'(x) = 9c_9 x^8 + 8c_8 x^7 + 7c_7 x^6 + 6c_6 x^5 + 5c_5 x^4 + 4c_4 x^3 + 3c_3 x^2 + 2c_2 x + c_1$$

```
double d[10];
int i;
for (i = 0; i < 9; i++)
   d[i] = (i + 1) * c[i + 1];
d[9] = 0.0;
```


Array Tradeoffs

Advantage.

Disadvantage.

Memory Address	107	108	109	110	111	112	113	114	115	116
Variable	c[0]	c[1]	c[2]	c[3]	c[4]	c[5]	c[6]	c[7]	c[8]	c[9]
Value	7.0	0.0	0.0	0.0	0.0	3.8	0.0	0.0	0.0	1.2

Array Example: Strings

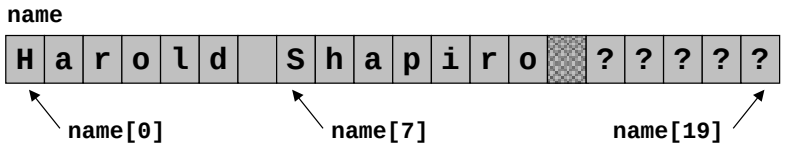
A variable of type `char` stores a character.

```
char c = 'H';
```

A STRING is an array of characters.

```
char name[20] = "Harold Shapiro";
```

- Implicitly ends with `'\0'` which is the same as 0.



Benford's Law

Examine listing of statistical data.

- Compute frequency count of LEADING DIGIT.
  - leading digit of 456789 is 4
- Print fraction of occurrences of each digit 1 - 9.
- What is distribution?

Use 10-element array count.

- `freq[i]` counts number of times `i` is leading digit.
- `tot` counts total number of items processed.
- Print `(freq[i] / tot)` for each `i`.

Benford's Law

benford.c

```
#include <stdio.h>

int leadingDigit(int x) {
   while (x >= 10)
      x /= 10;
   return x;
}

int main(void) {
   int i, d, x, tot = 0, freq[10] = {0};
   while (scanf("%d", &x) != EOF) {
      d = leadingDigit(x);
      freq[d]++;
      tot++;
   }
   for (i = 1; i < 10; i++)
      printf("%d: %f\n", i, 1.0 * freq[i] / tot);
   return 0;
}
```

compute leading digit of x

initialize all to zero

increment bin

Benford's Law

Newcomb (1881).

- Tables of logarithms.

Benford (1938).

- River area. Population.
  Newspaper. Specific heat.
  Pressure. Atomic weight.
  Drainage. Reader's Digest.
  Baseball. Black body.
  Death rates. Addresses.

  Scale invariant!

Hill (1996).

- Distribution of distributions.

Unix

```
% more princeton-files.txt
  96796
  4171208
  5830
  34343656
  ...

% gcc benford.c
% a.out < arizona-files.txt
  1: 0.3010
  2: 0.1761
  3: 0.1249
  4: 0.0969
  5: 0.0792
  6: 0.0669
  7: 0.0580
  8: 0.0512
  9: 0.0458
```

$$P_d = \ln\left(\frac{d+1}{d}\right) - \ln 10$$

Sorting

Goal: given N items, rearrange them in increasing order.

Applications.

- Sort a list of names.
- Find duplicates in a mailing list.
- Find the median.
- Identify statistical outliers.

name
Hauser
Hong
Hsu
Hayes
Haskell
Hanley
Hornet

name
Hanley
Haskell
Hauser
Hayes
Hong
Hornet
Hsu

Insertion Sort

Insertion sort.

- In ith iteration:
  - read ith value
  - repeatedly swap ith value with the one to its left if it is smaller

Property: after i^{th} iteration, array positions 0 through i contain original elements 0 through i in increasing order.

insertion-sort code fragment

```
for (i = 0; i < N; i++) {
   for (j = i; j > 0; j--)
      if (x[j-1] > x[j]) {
         swap = x[j]; x[j] = x[j-1]; x[j-1] = swap;
      }
}
```

Swap x[j] and x[j-1]

insertion-sort.c

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

int main(void) {
   int i, j;
   double swap, x[N];

   for (i = 0; i < N; i++)
      scanf("%lf", &x[i]);

   for (i = 0; i < N; i++) {
      for (j = i; j > 0; j--)
         if (x[j-1] > x[j]) {
            swap = x[j]; x[j] = x[j-1]; x[j-1] = swap;
         }
   }

   for (i = 0; i < N; i++)
      printf("%f\n", x[i]);
   return 0;
}
```

read input

print results

Array Function Example: Shuffling

Goal: shuffle n-element array.

- In ith iteration:
  - choose random integer r between 0 and i
  - swap values in positions r and i
- Need random access to arbitrary element ⇒ use arrays.

Property: after i^{th} iteration, array positions 0 through i contain random permutation of elements 0 through i.

shuffle.c

```
void shuffle(double a[], int n) {
    int i, r;
    double swap;
    for (i = 0; i < n; i++) {
       r = randomInteger(i+1);
       swap = a[r]; a[r] = a[i]; a[i] = swap;
    }
 }
```


Bicycle Problem

Bicycle problem.

- N kids go to a party and dump bicycle in a pile.
- Kids are blindfolded, and each one selects a bike at random.
- What is likelihood that at least one gets their own bike?

1964 Spacelander.

1937 Columbia Superb

1999 Schwinn Grape Krate

A Helper Function

Create a random permutation of integers 0 through n-1.

- Fill up array with elements 0 through n-1.
- Shuffle the array.

randomPermutation( )

```
void randomPermutation(int a[], int n) {
   int i;
   for (i = 0; i < n; i++)
      a[i] = i;
   shuffle_int(a, n);
}
```

Bicycle Problem

One simulation:

- Select a random permutation of N elements.
- Check to see if any value matches it index.

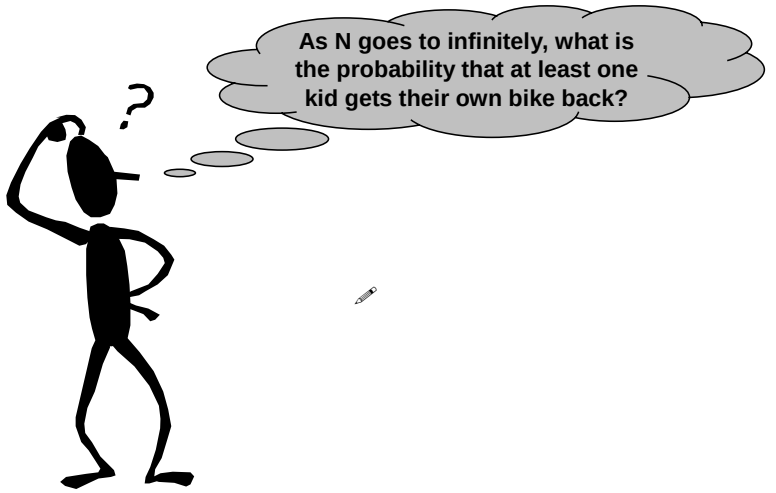
`(a[j] == j)` if j gets their own bike

bike.c

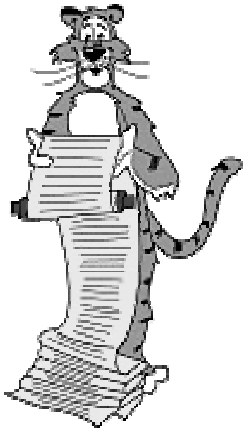
```
#define N      10000
#define TRIALS  1000

int main(void) {
  int i, j, cnt = 0, a[N];

  for (i = 0; i < TRIALS; i++) {
    randomPermutation(a, N);
    for (j = 0; j < N; j++)
      if (a[j] == j) {
        cnt++;
        break;
      }
  }

  printf("successes prob = %f\n",
     1.0 * cnt / TRIALS;
  return 0;
}
```

once you find a match, break out of loop

Bicycle Problem



Lecture P2: Supplemental Notes



LFBSR Revisited

All the b variables behave the same. Why not bundle together?

lfbsr.c

```
#include <stdio.h>
#define N 100

int main(void) {
  int i, new;
  int b10 = 0, b9 = 1, b8 = 1, b7 = 0, b6 = 1, b5 = 0;
  int b4  = 0, b3 = 0, b2 = 0, b1 = 1, b0 = 0;

  for (i = 0; i < N; i++) {
    new = b3 ^ b10;
    b10 = b9; b9 = b8; b8 = b7; b7 = b6; b6 = b5;
    b5  = b4; b3 = b2; b2 = b1; b1 = b0; b0 = new;
    printf("%d", new);
  }
  return 0;
}
```

LFBSR Revisited

All the b variables behave the same. Why not bundle together?

lfbsr-array.c

```
#include <stdio.h>
#define N   100
#define BITS 10

int main(void) {
  int i, j, new;
  int b[] = {0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0};

  for (i = 0; i < N; i++) {
    new = b[3] ^ b[10];
    for (j = BITS; j >= 1; j--)
       b[j] = b[j-1];
    b[0] = new;
    printf("%d", new);
  }
  return 0;
}
```


Array Example: Yahtzee

Yahtzee is a "fast-paced, tension-filled game for one or more players" that involves rolling dice. (Milton Bradley)

- Throw five dice. If they all match you yell "YAHTZEE".
- See `www.yahtzee.com` for other rules.

```
int dice[5];
int i, match;

for (i = 0; i < 5; i++)
   dice[i] = randomInteger(6) + 1;

match = 1;
for (i = 1; i < 5; i++)
   if (dice[i] != dice[0])
      match = 0;

if (match == 1)
   printf("YAHTZEE!\n");
```

Array Example: Computing a Histogram

Unix

```
% more histo.data
  68 69 67 65 60 68 67 94 59 54 21 96 95 94 55 69 93
  64 65 93 54 64 94 63 65 89 93 28 92 62 61 87 92 55
  89 51 95 85 88 94 86 93 93 84 86 93 84 92 93 92 85
  90 84 87 88 90 85 87 91 87 87 82 85 85 88 87 86 81
  90 85 81 85 84 78 90 85 79 91 92 75 78 89 76 91 91
  75 78 89 76 90 81 74 78 80 76 80 81 84 76 78 74 79
  79 84 74 77 84 78 76 82 70 77 83 76 74 70 83 76 81
  72 73 67 81 72 69 84 83 71 72 84 73 82 83 70 71 82
  69 82 82 73 80
% gcc histo.c
% a.out < histo.data
   0- 9
  10-19
  20-29 **
  30-39
  40-49
  50-59 ******
  60-69 ******************
  70-79 **********************************
  80-89 ***************************************************
  90-99 ****************************
```


Array Example: Computing a Histogram

Histogram = bar graph of number of occurrences of each value.

- Grades in range 0-99.
- How many in ranges 0-9, 10-19, 20-29, . . ., 90-99?

Can do without arrays with 50+ lines of repetitive code.

Elegant with arrays.

- Use data as index.
- See Program 3.7 in Sedgewick.

Array Example: Computing a Histogram

histo.c

```
#include <stdio.h>
int main(void) {
   int i, j, val, bin;
   int h[10];

   for (i = 0; i < 10; i++)
      h[i] = 0;

   while (scanf("%d", &val) != EOF) {
      bin = val / 10;
      h[bin]++;
   }

   for (j = 0; j < 10; j++) {
      printf("%2d-%2d ", j*10, j*10+9);
      for (i = 0; i < h[j]; i++)
         printf("*");
      printf("\n");
   }
   return 0;
}
```

initialize array

calculate which bin

increment that bin

print right number of stars for each bin

Array Example: The Birthday Problem

People enter an empty room until a pair of people share a birthday.
How long will it take on average?

- Assume birthdays are uniform random integers between 0 and 364.

b[d] = 0 if day d not "filled"
= 1 if day d is "filled"

bday( )

```
#define DAYS 365

int bday(void) {
  int i, d, b[DAYS];
  for (i = 0; i < DAYS; i++)
     b[i] = 0;
  for (i = 0; i <= DAYS; i++) {
     d = randomInteger(DAYS);
     if (b[d] == 1)
        return i;
     else
        b[d] = 1;
  }
}
```

Mark all birthdays as not "filled".

If birthday d is "filled" return; o/w as "filled."

Array Example: The Birthday Problem

People enter an empty room until a pair of people share a birthday.
How long will it take on average?

- Assume birthdays are uniform random integers between 0 and 364.

bday.c

```
#include <stdio.h>
#include <stdlib.h>
#define DAYS   365
#define TRIALS 100

int randomInteger(int n) { ...}
int bday(void) { ... }

int main(void) {
  int i;
  for (i = 0; i < TRIALS; i++)
    printf("%d\n", bday());
  return 0;
}
```

run simulation several times