Lecture P2: Arrays

Arrays

Built-in to C.
- Declare using 

\[ \text{double } a0, a1, a2, a3, a4, a5, a6, a7, a8, a9; \]

vs.

\[ \text{double } a[10]; \]

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

- Caveats:
  - Array indices start at 0 not 1.
  - "Ghastly error" to access element 10 of a 10 element array.

### Array Example: Manipulate Polynomials

C representation of \( 1.2x^9 + 3.8x^5 + 7.0 \).

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

Double \( a[10]; \)

Possible memory representation (assuming array starts at 107).

<table>
<thead>
<tr>
<th>Memory Address</th>
<th>107</th>
<th>108</th>
<th>109</th>
<th>110</th>
<th>111</th>
<th>112</th>
<th>113</th>
<th>114</th>
<th>115</th>
<th>116</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>7.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>3.8</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.2</td>
</tr>
</tbody>
</table>

Initialize variables before using

Evaluating \( p(x) \) and \( x = 3.14 \).

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

Clever alternative (Horner’s method).

\[
\text{double } x = 3.14, p = 0.0; \\
\text{int } i; \\
\text{for } (i = 0; i < 10; i++) \\
\quad p += c[i] * \text{pow}(x, i);
\]

\[
\text{double } x = 3.14, p = 0.0; \\
\text{int } i; \\
\text{for } (i = 0; i < 10; i++) \\
\quad p += c[i] + (x * p);
\]
**Array Example: Manipulate Polynomials**

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

Differentiating.

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

```c
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.**

- Can get to each item quickly.

**Disadvantage.**

- Consumes space for unused items.

<table>
<thead>
<tr>
<th>Memory Address</th>
<th>107</th>
<th>108</th>
<th>109</th>
<th>110</th>
<th>111</th>
<th>112</th>
<th>113</th>
<th>114</th>
<th>115</th>
<th>116</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>7.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>3.8</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.2</td>
</tr>
</tbody>
</table>

---

**Array Example: Strings**

A variable of type `char` stores a character.

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

A STRING is an array of characters.

```c
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 \((\text{freq}[i] / \text{tot})\) for each i.
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!

- Distribution of distributions.

Unix

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

% gcc benford.c
% a.out < arizona-files.txt

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

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 ith 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;
        }
}
```
# 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;
}

int i, j;
double swap, x[N];
Bicycle Problem

One simulation:
- Select a random permutation of N elements.
- Check to see if any value matches its index.

```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;
}
```

Lecture P2: Supplemental Notes

```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; b4 = b3; b3 = b2; b2 = b1; b1 = b0; b0 = new;
    printf("%d", new);
}
```

LFBSR Revisited

All the b variables behave the same. Why not bundle together?
**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](http://www.yahtzee.com) for other rules.

```c
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**

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

```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;
}
```

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.

```c
#include <stdio.h>
#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;
  }
  return 0;
}
```