Calculate $x^y$

Algorithm: repeated multiplication

\[ x \times x \times x \times \cdots \times x \]

y times

<table>
<thead>
<tr>
<th>y</th>
<th>x</th>
<th>i</th>
<th>p=x^i</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>4</td>
<td>81</td>
</tr>
</tbody>
</table>

// Pre: y ≥ 0
// Returns $x^y$
int power(int x, int y) {
    int p = 1;
    int i = 0;
    while (i < y) { // Repeat y times
        p = p*x;
        i = i + 1;  // p = x^i
    }
    return p;
}
Calculate $x^y$

```c
int power(int x, int y) {
    int p = 1;
    for (int i = 0; i < y; i = i + 1) {
        p = p * x;
    }
    return p;
}
```

// Pre: $y \geq 0$
// Returns $x^y$

// Up-down sequence

- Write a program that reads a positive integer $n$ and prints an up-down sequence (one number per line)

- Example ($n=6$):

```c
int factorial(int n) {
    int f = 1;
    for (int i = 1; i <= n; ++i) f = f * i;
    return f;
}
```

```
  1 2 3 4 5 6 5 4 3 2 1
```
// This program reads a positive integer (n)
// and prints an up-down sequence
// (one number per line).
// Example: 1 2 3 ... n-1 n-1 ... 3 2 1

int main() {
    int n;
    cin >> n;

    // up sequence: 1 2 3 ... n-2 n-1 n
    for (int i = 1; i <= n; ++i) cout << i << endl;

    // down sequence: n-1 n-2 ... 3 2 1
    for (int i = n - 1; i > 0; --i) cout << i << endl;
}

// This program reads a positive integer (n)
// and prints an up-down sequence
// (one number per line).
// Example: 1 2 3 ... n-1 n-1 ... 3 2 1

int main() {
    int n;
    cin >> n;

    // up sequence: 1 2 3 ... n-2 n-1 n
    for (int i = 1; i <= n; ++i) cout << i << endl;

    // down sequence: n-1 n-2 ... 3 2 1
    for (int i = n - 1; i > 0; --i) cout << i << endl;
}

// This program reads three numbers and
// prints the same numbers in ascending order.

int main() {
    int x, y, z;
    cin >> x >> y >> z;

    if (x > y) {
        int t = x;
        x = y;
        y = t;
    } // We know that x <= y

    if (y > z) {
        int t = y;
        y = z;
        z = t;
    } // We know that z is the largest number

    if (x > y) {
        int t = x;
        x = y;
        y = t;
    } // We know that x <= y <= z

    cout << x << " " << y << " " << z << endl;
}

// This program reads two numbers and prints
// the minimum and the maximum (in this order).

int main() {
    int x, y;
    cin >> x >> y;

    if (x > y) {
        int min = y;
        int max = x;
    } else {
        int min = x;
        int max = y;
    }

    cout << min << " " << max << endl;
}

Up-down sequence

Same name, different variables

Summary

• A for loop is a special type of repetitive
  statement with a loop counter.

• It is naturally used when the number of
  iterations is known before entering the loop.

• Recommendations:
  – Declare the loop counter locally (when possible).
  – Update the loop counter by a constant (++i, --i).
  – Do not modify the loop counter inside the for loop.

Sort three numbers

Print min and max

min and max
are not visible

Wrong!
Print min and max

// This program reads two numbers and prints
// the minimum and the maximum (in this order).

int main() {
    int x, y;
    cin >> x >> y;
    int min, max;
    if (x > y) {
        min = y;
        max = x;
    } else {
        min = x;
        max = y;
    }
    cout << min << " " << max << endl;
}

Scopes and visibility

{ // a and b are not visible
    int a = 1, b = 20;
    // a and b are visible
    cout << a << endl;
    {
        // c is not visible, a and b are visible
        cout << a + b << endl;
        int b = 5, c = 4;
        // a, c and the inner b are visible,
        // but the outer b is not visible
        cout << a + b << endl;
        cout << c << endl;
    }
    // c is not visible
    cout << b << endl;
}

Summary

• Variables are only visible within their scope.

• Recommendations:
  – Declare and use variables as locally as possible.
  – When possible, initialize variables with their declaration.
  – Use meaningful names (e.g., min, count, avg, ...)

• Do not try to reuse variables: fewer variables does not imply less memory.

Drawing a rectangle

• Write a program that reads the dimensions of a rectangle (x, y) and prints x columns and y rows of asterisks.

• Example (x=8, y=5):

  **
  **
  **
  **
  **

Introduction to Programming © Dept. CS, UPC 13
Introduction to Programming © Dept. CS, UPC 14
Introduction to Programming © Dept. CS, UPC 15
Introduction to Programming © Dept. CS, UPC 16
// Draws a rectangle
// Reads the dimensions (x, y) of a rectangle
// and prints x columns and y rows of asterisks.
int main() {
    int x, y;
    cin >> x >> y;
    drawRectangle(x, y);
}

// Prints a rectangle with <ncols> columns and
// <nrows> rows of asterisks.
void drawRectangle(int ncols, int nrows) {
    for (int i = 0; i < nrows; ++i) {
        printRow(ncols);
    }
}

// Prints a row of n asterisks.
void printRow(int n) {
    for (int i = 0; i < n; ++i) cout << "\*";
    cout << endl;
}

// Reads the dimensions (x, y) of a rectangle
// and prints x columns and y rows of asterisks.
int main() {
    int x, y;
    cin >> x >> y;
    for (int r = 0; r < y; ++r) {
        for (int c = 0; c < x; ++c) cout << "\*";
        cout << endl;
    }
}
• Write a program that reads an integer \( n \) and prints a right equilateral triangle with the length of the cathetus being \( n \).

• Example (n=7):

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

```
// Reads an integer \( n \) and prints a right equilateral triangle with the length of the cathetus being \( n \).
int main() {
    int n;
    cin >> n;
    for (int r = 1; r <= n; ++r) {
        for (int c = 0; c < r; ++c) cout << "*";
        cout << endl;
    }
}
```
Exercises

Draw the following shapes (for any n):

- **:**
- *:* *

Chess board

Summary

- Nested loops are useful when treating multi-dimensional data, e.g., rows/columns, height/width, matrices, etc.

- Recommendations:
  - Use for loops if the number of iterations is known before entering the loop.
  - Use a different local counter for each loop.