Introduction to Programming (CS 101)

Spring 2024

Lecture 12:

Recursion HW, Introduction to arrays, Midsem recap

Instructor: Preethi Jyothi

Based on material developed by Prof. Abhiram Ranade

Common C++ Errors

CS 101, 2025

Some common C++ errors (based on concepts studied so far)

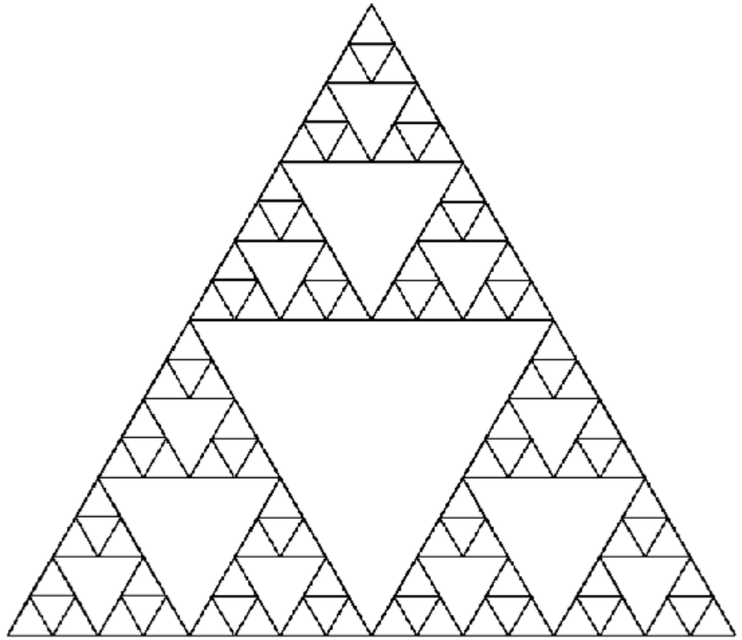
- Using = (assignment) instead of == (equal to)
- Not enclosing a multiple-statement body within parenthesis `{ }`
- Variable scope issues
- Dangling else
- Integer division when floats/doubles are required
- Operator precedence; use parentheses whenever precedence is unclear
- Lossy type casting. E.g., `int i = 32; bool b = i;`
- Loop limits being off by one (e.g., using `'i = 0; i <= n'` instead of `'i = 0; i < n'` for `n` iterations)
- Division by zero
- Sequentially address compiler errors; errors could compound

Recursion problems

CS 101, 2025

Homework Exercises

Write a recursive program to draw the following fractal with repeating equilateral triangles



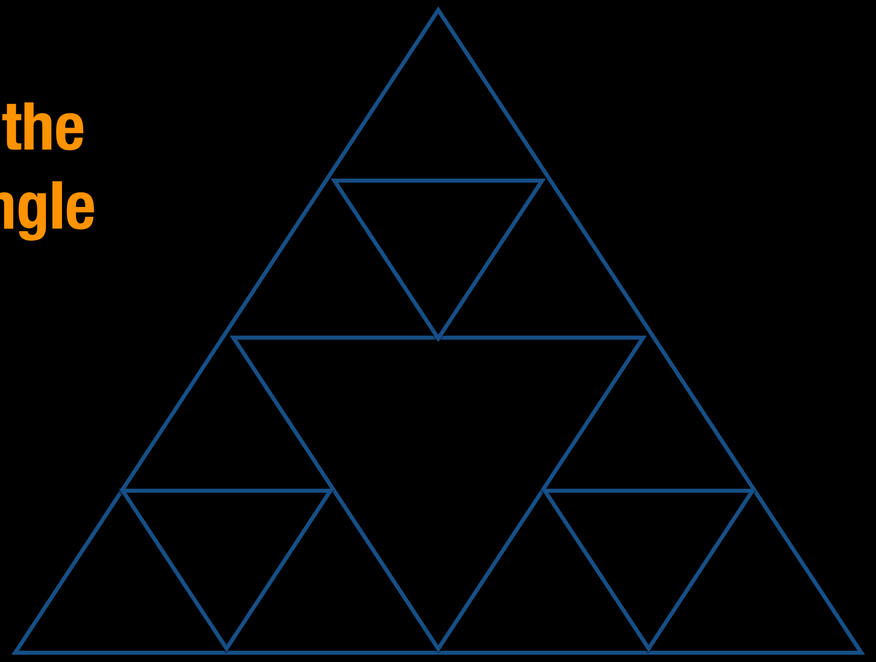
Write a recursive function to print out all permutations of a string. Assume the string is a single word. If there are duplicates, you can print them all out.

If the input is "`out`", your code prints:

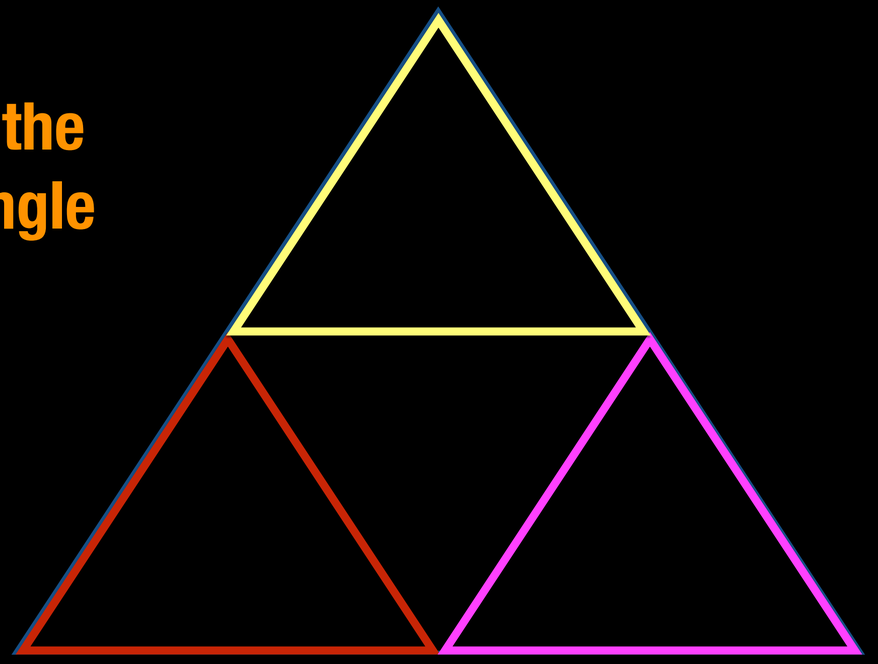
```
out
otu
uot
uto
tou
tuo
```

Fractal: https://en.wikipedia.org/wiki/Sierpi%C5%84ski_triangle

Unpacking the fractal triangle



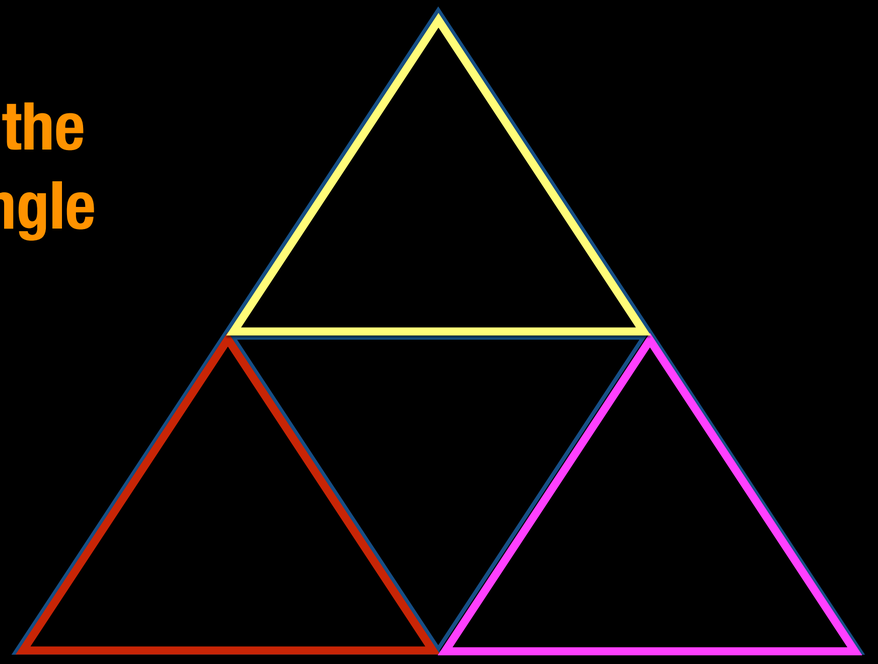
Unpacking the fractal triangle



```
void dtriangle(double side, int level) {
  for(int i = 0; i < 3; i++) {
      forward(side); left(120);
  }
  if(level == 0) return;
  ⋮
```

```
 if (level == 1) {
    for(int i = 0; i < 3; i++) {
      forward(side/2); left(120);
    }
    penUp(); forward(side/2); penDown();
    for(int i = 0; i < 3; i++) {
      forward(side/2); left(120);
    }
    penUp(); left(120); forward(side/2);
    right(120); penDown();
    for(int i = 0; i < 3; i++) {
      forward(side/2); left(120);
    }
    penUp(); right(120); forward(side/2);
    left(120); penDown();
 }
}
```

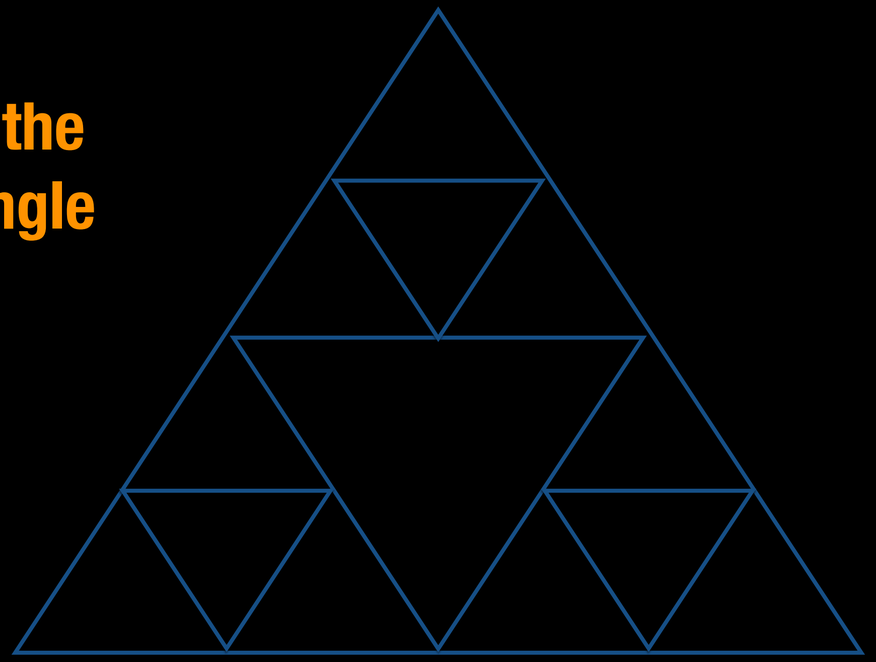
Unpacking the fractal triangle



```
void dtriangle(double side, int level) {
  for(int i = 0; i < 3; i++) {
      forward(side); left(120);
  }
  if(level == 0) return;
  ⋮
```

```
 if (level == 1) {

      dtriangle(side/2, 0);

    penUp(); forward(side/2); penDown();

      dtriangle(side/2, 0);

    penUp(); left(120); forward(side/2);
    right(120); penDown();

      dtriangle(side/2, 0);

    penUp(); right(120); forward(side/2);
    left(120); penDown();
 }
}
```

Unpacking the fractal triangle



```
 if (level == 1) {

      dtriangle(side/2, 0);

    penUp(); forward(side/2); penDown();

      dtriangle(side/2, 0);

    penUp(); left(120); forward(side/2);
    right(120); penDown();

      dtriangle(side/2, 0);

    penUp(); right(120); forward(side/2);
    left(120); penDown();
 }
}
```

Print all permutations of a string using a recursive function

```
void permute(string& str, int l, int r) {
   if(l == r)
      cout << str << endl;  //Base case
   else {
    for(int i = l; i <=r; i++) {
      swap(str[l], str[i]);
      permute(str, l+1, r);
      swap(str[l], str[i]);
    }
   }
}
```

Introduction to Arrays

CS 101, 2025

Example1: Counting vowels

- We want to read a word (in lowercase) as input and maintain counts of each vowel

```
int main() {
  int Na=0, Ne=0, Ni=0, No=0, Nu=0; char c;
  for(cin >> c; c >= 'a' && c <= 'z'; cin >> c) {
    if (c == 'a') ++Na;
    else if (c == 'e') ++Ne;
    else if (c == 'i') ++Ni;
    else if (c == 'o') ++No;
    else if (c == 'u') ++Nu;
  }
  cout << "#a = " << Na << ", #e = " << Ne << ", #i = " << Ni
                     << ", #o = " << No << ", #u = " << Nu << endl;
}
```

What if we want to count the occurrences of every letter in the alphabet?

26 counters! 😱

Example2: Sorting

- Read 3 numbers as input and print them in ascending order

```
int main() {
  int a1, a2, a3;
  cin >> a1 >> a2 >> a3;
  if (a1 <= a2 && a2 <= a3)
     cout << a1 << ", " << a2 << ", " << a3 << endl;
  else if (a1 <= a3 && a3 <= a2)
     cout << a1 << ", " << a3 << ", " << a2 << endl;
  else if (a2 <= a3 && a3 <= a1)
     cout << a2 << ", " << a3 << ", " << a1 << endl;
  else if (a2 <= a1 && a1 <= a3)
     cout << a2 << ", " << a1 << ", " << a3 << endl;
  else if (a3 <= a1 && a1 <= a2)
     cout << a3 << ", " << a1 << ", " << a2 << endl;
  else if (a3 <= a2 && a2 <= a1)
     cout << a3 << ", " << a2 << ", " << a1 << endl;
}
```

What if we want to sort n numbers?

Will need n! checks 😱

Arrays

- Arrays let us declare and use a large number of variables (of the same type)
- Syntax:

```
data-type array-name [ number ] ;
```

- Example: `int N[26];`
- Each variable in the array is referred to using an index: e.g., `N[i]`
- The first element is `N[0]` and the last element is `N[`number `- 1]`

Counting letters of the alphabet using an array

```
int N[26];
for(int i = 0; i < 26; i++)
  N[i] = 0;
char c;
for(cin >> c; c >= 'a' && c <= 'z'; cin >> c)
  N[c-'a']++;
for(char c = 'a'; c <= 'z'; c++)
  cout << "#" << c << " = " << N[c-'a'] << endl;
```

Accessing array elements

- Setting values during initialization:
  - Explicitly: `int N[3] = {-1, 0, 1};`
  - Can initialize a few elements, and have the rest all set to `0`
    - `int N[26] = {1}; // N[0] = 1, N[i] = 0 for all i from 1 to 25`
    - `int N[26] = { }; // all values are set to 0`
- At any point (not just initially), we can assign values one by one. E.g., `N[i] = i;`
- Cannot assign to an array variable itself
    - `int N[26];`
    - `N = N; //error: array type is not assignable`

Counting vowels

- To read in a word (lower case) and count the vowel occurrences

```
int main() {
  char c, vowels[] = {'a', 'e', 'i', 'o', 'u'};
  int N[5] = {};
  for(cin >> c; c >= 'a' && c <= 'z'; cin >> c)
    for (int i=0; i<5; i++)
      if(c == vowels[i]) {
        N[i]++; break;
      }
  for (int i=0; i<5; i++)
    cout << "#" << vowels[i] << " = " << N[i] << (i==4?"":", ");
  cout << endl;
}
```

Array bounds

- It is the programmer's responsibility (rather than the compiler's) to ensure that when accessing an array, the index remains within bounds (i.e., 0 ≤ index ≤ number of elements -1)

- E.g., the compiler will not complain/warn about this:

  - `int X[10], n;  cin >> n;  X[n] = 1;`

- Instead include a bound-check: `if (n>=0 && n<10)  X[n]=1; else …`

  - Or enforce index bounds in the program logic (e.g., `X[abs(n)%10]`)

- If the array index is out of bounds, it can cause the program to behave in unspecified ways (possibly crash, or access other variables). (Remember the Crowdstrike bug?)

https://en.wikipedia.org/wiki/2024_CrowdStrike-related_IT_outages

Palindrome check using arrays

- Read an n-letter word and check if it is a palindrome

```
int main() {
  const int Nmax = 100; //array size has to be known at compile time
  char text[Nmax];
  int n; cin >> n;
  if(n > Nmax) { cout << "Too long\n"; return -1; }
  for (int i=0; i<n; i++)  cin >> text[i];
  for (int i=0; i<n/2; i++) {
    if (text[i] != text[n-1-i]) {
      cout << "Not a palindrome!\n";  return 1;
    }
  }
  cout << "Palindrome!\n";
}
```

Midsem Recap

CS 101, 2025

Midsem Recap (IA)

Match `(A)` with one of `(i), (ii), (iii)` which is equivalent

```
(A)   if(a > 10 && b > 10) f();
(B)   if(a > 10 || b > 10) f();

(i)   if (a > 10) f(); else if (b > 10) f();
(ii)  if (a > 10) f(); if (b > 10) f();
(iii) if (a > 10) if (b > 10) f();
```

i

ii

iii

Midsem Recap (IB)

Match `(B)` with one of `(i), (ii), (iii)` which is equivalent

```
(A)   if(a > 10 && b > 10) f();
(B)   if(a > 10 || b > 10) f();

(i)   if (a > 10) f(); else if (b > 10) f();
(ii)  if (a > 10) f(); if (b > 10) f();
(iii) if (a > 10) if (b > 10) f();
```

- ☑ i
- ☐ ii
- ☐ iii

Midsem Recap (IIA)

Say we want to set the `j`th least significant bit of a number to `0`. For example, if the binary representation of a `short int num` is `0000000000101101` and `j=3`, `num` should become `0000000000101001`

We want to do this in a single statement using bitwise operators. Does the following expression achieve the desired outcome? Explain why or why not.

```
num ^= (num & (1 << (j-1)));
```

A Yes

B No

Midsem Recap (IIB)

Say we want to set the `j`th least significant bit of a number to `0`. For example, if the binary representation of a `short int num` is `0000000000101101` and `j=3`, `num` should become `0000000000101001`

We want to do this in a single statement using bitwise operators. Does the following expression achieve the desired outcome? Explain why or why not.

```
num &= ~(1 << (j-1));
```

A Yes

B No

Midsem Recap (IIC)

Say we want to set the `j`th least significant bit of a number to `0`. For example, if the binary representation of a `short int num` is `0000000000101101` and `j=3`, `num` should become `0000000000101001`

We want to do this in a single statement using bitwise operators. Does the following expression achieve the desired outcome? Explain why or why not.

```
num |= (1 << (j-1));
```

A Yes

B No

Midsem Recap (III)

What is the output?

```
bool istrue() {
 cout << "true\n";
 return true;
}

bool isfalse() {
 cout << "false\n";
 return false;
}

int main() {
if(isfalse() || !!(true && !isfalse()))
  cout << "true\n";
}
```

A
```
false
false
```

B
```
false
false
true
```

C
```
false
```

D
```
false
true
```

Midsem Recap (IV)

What do each of the following statements output?

```
char c1 = 'A', c2 = '6';  //ASCII codes of '6','A','a': 54,65,97
```

```
char c = c1 + c2 - '0'; cout << c;
```

G output

```
char c = c1 + c2; cout << c;
```

w output

```
char c = c1 + int(c2); cout << c;
```

w output

```
char c = c1 + (1<<1) + (1<<2); cout << c;
```

G output

```
char c = c1 + 0xF-9; cout << c;
```

G output

Midsem Recap (V)

Fill in the blank below to find the largest digit in a given non-negative number. Note `max(x, y)` returns the maximum of integers `x` and `y`.

```
int findLargestDigit(int n) {
  int md = 0;
  while(n > 0) {
    md = [                ];
    n /= 10;
 }
  return md;
}

int main() {
  int n;
  cin >> n;
  cout << findLargestDigit(n) << endl;
}
```

Midsem Recap (V)

Fill in the blank below to find the largest digit in a given non-negative number. Note `max(x, y)` returns the maximum of integers `x` and `y`.

```
int findLargestDigit(int n) {
  int md = 0;
  while(n > 0) {
    md = max(md, n%10);
    n /= 10;
 }
  return md;
}

int main() {
  int n;
  cin >> n;
  cout << findLargestDigit(n) << endl;
}
```