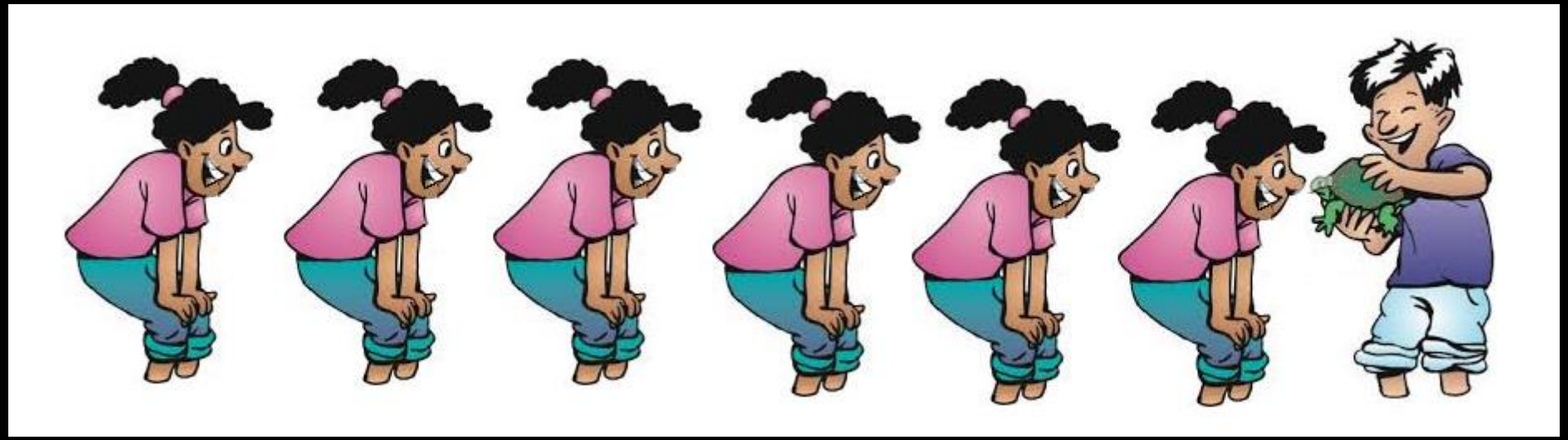
Getting Ready for Canadian Computing Competition

Class 4
December 21, 2019

Outline

- Recursion
- Structures
- Pointers

Recursion



Recursion

- A function calls itself repeatedly until a base case is encountered

- A recursive function always contains:
  - A base case (when the parameter reaches a certain value)
  - A recursive call (falling the function itself with updated parameters)

Example

- Factorial function: f(n) = n*f(n-1)

Example

- Factorial function: f(n) = n*f(n-1)
- base case: if n<=1 then f(n) = 1
- Recursive call: n * f (n – 1)

Exercise

Write a function, find_max (vector<int> lst, int n) that finds the maximum element in the vector lst with length n.

If the main function is as follows:

```
int main() {

    vector <int> Numbers = {1,2,3,89, 6,2, 0, 98, 2, 3};

    cout << find_max (Numbers, 6);

}
```

The number 98 should be printed.

Exercise

Given an array of length N and an integer x, return the first index of the first occurrence of x. Return -1 if x is not found. Use recursion.

The first line of input is N, x, separated by a space, followed by N lines of integers.

Exercise

Recursion may be optional for some questions, but not all of them…

A child is running up a staircase with N steps. They can hop 1 step, 2 steps or 3 steps at a time. Count how many possible ways the child can run up to the stairs.

Sample input:
3
Sample output:
4

Explanation:
Method 1: Hop 3 steps
Method 2: Hop 2 steps, then 1 step
Method 3: Hop 1 step, then 2 steps
Method 3: Hop 1 step, then 1 step, then 1 step

Practice (hw)

CCC 2016 S2

Structures

- Given a 2D coordinate system with points p1 = (x1, y1), p2 = (x2, y2), we want to write a function that adds p1 and p2 to return the point (x1 + x2, y1 + y2).
- How do we return two integers in one function?
- Array?
  - arr[0] -> x
  - arr[1] -> y

Structures

Creating a Structure

Syntax:

```
struct structureName{
    member1Type member1Name;
    member2Type member1Name;
    …
    membernType membernName;
};
```

Example

- Back to the coordinates problem…
- Given a 2D coordinate system with point p = (x, y), we want to write a function that moves p a to the left and b to the right. The updated p is p = (x + a, y + b).
- We can declare a structure:

```
struct Point {
   int x, y;
};
```

Example Continued…

- We reference the elements inside the structure using dot notation (.).
- Syntax: variableName.elementName

Using the structure:

```
int main() {
    Posn p1; // declare it
    p1 = {3,4}; // unlike arrays, this is allowed, not recommended
    Posn p2 {1,2}; // declaration + initialization
    cout << "(" << p1.x << "," << p1.y << ")" << endl; // (3,4)
}
```

Example Continued…

We write a function to add p1 and p2:

```
Posn add_two_points (Posn p1, Posn p2) {
    return Posn {p1.x + p2.x, p1.y + p2.y};
}
```

Example Continued…

We can also write a function to print (x, y) coordinates. Write a function , print_posn, that prints a position in the format (x,y).

Example Continued…

Now our main function is:

```
int main() {
    Posn p1 {3,4};
    Posn p2 {1,2};
    Posn result = add_two_points(p1, p2);
    print_posn (p1);
    print_posn (p2);
    print_posn (result);
}
```

Practice

The student class:

Initialize a structure of students that contain the following information:

name, student number, grade

Practice

The student class:

Initialize a structure of students that contain the following information:

name, student number, grade

```
struct Student {
    string name;
    int studentNum, grade;
};
```

Practice

Write a function, compare_students, that takes two parameters which are two students and returns true if the first student is “higher” than the second. A student, s1, is higher than another student, s2, if s1 has a higher grade. If there is a tie, s1 is higher when it has a higher student number.

Practice

Write a function, compare_students, that takes two parameters which are two students and returns true if the first student is “higher” than the second. A student, s1, is higher than another student, s2, if s1 has a higher grade. If there is a tie, s1 is higher when it has a higher student number.

```
bool compare_students(Student s1, Student s2) {
    if (s1.grade != s2.grade) {
        return (s1.grade > s2.grade)? true: false;
    }
    else {
        return (s1.studentNum > s2.studentNum)? true : false;
    }
}
```

Practice

Write a function that prints a vector of students as follows:

Name student_number grade

```
void print_students (vector <Student> S) {
    for (auto &i: S) {
        cout << i.name << " " << i.studentNum << " " << i.grade
<< endl;
    }
}
```

Practice

Modify bubble sort to order the vector of students from highest to lowest. (hint: use compare_students as a helper function).

Practice

Modify bubble sort to order the vector of students from highest to lowest.

```
void sortStudents (vector<Student> &S) {
    int n = S.size();

    for (int i = 0; i < n - 1; ++i) {
        for (int j = 0; j < n - i - 1; ++j) {
            if (!compare_students(S[j], S[j+1])) {
                swap (S[j], S[j+1]);
            }
        }
    }
    print_students (S);
}
```

Practice

Write the main function that reads in n students where each student takes 3 lines, name, student number, and grade. Store them in a vector and print a sorted version of the students

Practice

Write the main function that reads in n students where each student takes 3 lines, name, student number, and grade. Store them in a vector and print a sorted version of the students

```
int main() {
    int n;
    cin >> n;
    string name;
    int stuNum, grade;
    vector<Student> Students;
    for (int i = 0; i < n; ++i) {
        cin >> name;
        cin >> stuNum;
        cin >> grade;
        Students.emplace_back(Student{name, stuNum, grade});
    }
    sortStudents(Students);
    print_students (Students);
}
```

Nested Structures (hw)

1. Define another structure, Date, that contains three integers: year, month, and day
2. Add another field to the student structure called birthday
3. Write a function, compare_birthdays, that compares two birthdays, b1 and b2, and returns true if the first is earlier than the second. Return false otherwise.
4. Write a function, sortStudents_birthday, that sorts a vector of students from oldest to youngest

Pointers

Pointers

- variables are locations in computer memory which can be accessed by their identifier (variable name)
- Each variable can be located in the memory by its address

Address-of operator (&)

- address-of operator (&) obtains the address of a variable
- address-of operator (&) is applied to whatever precedes it
- Example:

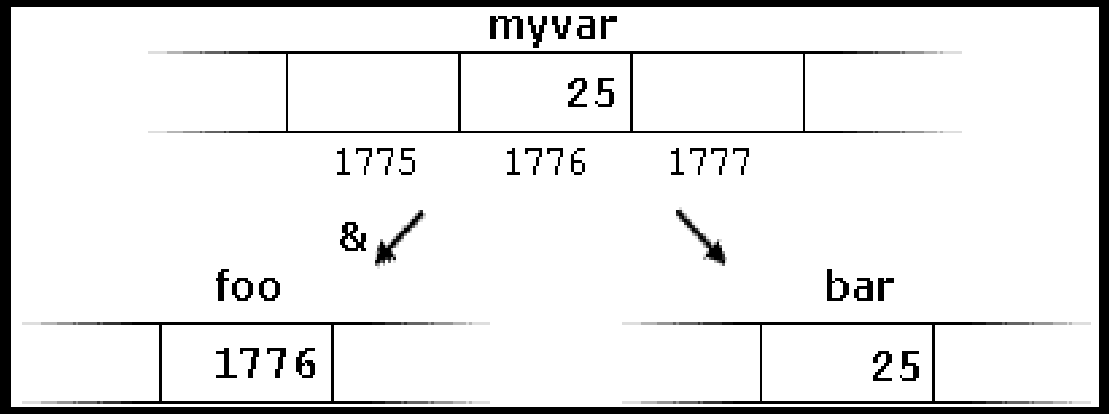
foo = &myvar;

foo now contains the address of myvar

Example of Address-of Operator (&)

assume that myvar has memory address 1776:

myvar = 25;
foo = &myvar;
bar = myvar;



Dereference operator (*)

- We can use a pointer to access the variable they point to
- dereference operator (*) means "value pointed to by“
- It will get the value of whatever pointer precedes it
- If what precedes is not a pointer, ERROR (syntax error)

- Ex). fooVal = *foo;

// "fooVal equal to value pointed to by foo where foo is a pointer”

fooVal = foo; // fooVal is a pointer

fooVal = *foo; // fooVal equal to value pointed to by foo (25)

Pointer Operator Summary

- & is the address-of operator, and can be read simply as "address of“
- * is the dereference operator, and can be read as "value pointed to by“
- They are complementary: an address obtained with & can be dereferenced with *

Exercise

Assume that the following variables have been declared, what are their types?

myvar = 25;

foo = &myvar;

A. myvar -> int, foo -> int pointer

B. myvar -> int, foo -> int

C. myvar -> int pointer, foo -> int pointer

Exercise

Given the previous declaration with myvar stored at memory address 1234, are these statements true or false?

```
myvar = 25;
foo = &myvar;
```

1. &myvar = 1234;
2. myVar == &1234;
3. foo == 1234
4. *foo == myvar
5. *(&myvar) == myvar
6. &(*myvar) == myvar
7. &(*foo) == foo

A. True
B. False

Declaring Pointers

- syntax: type * name;
- where type is the data type that the pointer points to

- Example

Int * num_p; // int pointer

Char * char_p; // char pointer

- Although they point to different data types that have difference sizes, they are just pointers
- All pointers take up 4 bytes of memory

Exercise

What will the following print? (Note: sizeof (int) == 4, sizeof (char) == 1)

```
int* num_p;
char *char_p;
vector <int> vec = {1,2,3};
vector <int> *vec_p = &vec;
cout <<sizeof (num_p) << endl;
cout <<sizeof (char_p) << endl;
cout <<sizeof (vec) << endl;
cout <<sizeof (vec_p) << endl;
```

Summary of *

- * is used for
  - Pointer declaration
  - Deference operator

They are not the same thing!

Example

We can change the value of an integer implicitly using pointers.

```
int num = 10;
cout << "before: " << num << endl;
int * p = &num;
*p = 20;
cout << "after: " << num << endl;
```

This prints:

```
10
20
```

Exercise

What does this print?

```
int num1 = 30, num2 = 20;
cout << "before: " << num1 << " " << num2 << endl;
int * p1, *p2;
p1 = &num1;
p2 = &num2;
*p1 = 10;
*p2 = *p1;
cout << "after: " << num1 << " " << num2 << endl;
```

Exercise

What does this print?

```
int num1 = 30, num2 = 20;
cout << "before: " << num1 << " " << num2 << endl;
int * p1, *p2;
p1 = &num1;
p2 = &num2;
*p1 = 10;
*p2 = *p1;
cout << "after: " << num1 << " " << num2 << endl;
```

before: 30 20

after: 10 10

Exercise

Note the difference between

```
int num1 = 30, num2 = 20;
cout << "before: " << num1 << " " << num2 << endl;
int * p1, *p2; // and int *p1, p2;
p1 = &num1;
p2 = &num2;
*p1 = 10;
*p2 = *p1;
cout << "after: " << num1 << " " << num2 << endl;
```

Null Pointers

- pointers are meant to point to valid addresses
- uninitialized pointers can point to unknown places
- Example

```
int *p;
cout << *p << endl; // you don't know what this will print
```

- Accessing this pointer causes undefined behavior
- If you want a pointer to point to nowhere, we use the null pointer (0 or nullptr)

```
int *p = nullptr; // recommend nullptr over 0
int *q = 0;
```

Null Pointers

- You can compare pointers with nullptr
- Example

```
int *p;
int *q = nullptr;
if (p == nullptr) {
    cout << "p is null" << endl;
}
if (q == nullptr) {
    cout << "q is null" << endl;
}
```

This prints:
q is null

CCC 2016 S3