Practice test for midterm 2

September 17, 2020

1 Functions

► Write a function which takes in two `int` parameters and returns their *average*. (Remember that if a function takes in parameters, it does not need to use `cin`, and if it returns a value it does not need `cout`.) Write the implementation (definition) of this function, write its declaration, and write an example of a function call using this function.

► Write a function which prompts the user to enter a *positive* (> 0) integer and which returns the value the user entered. If the user does not enter a positive integer, the function should return `0`.

► Write a function which prompts the user to enter a *positive* (> 0) integer and which returns the value the user entered. If the user does not enter a positive integer, the function should use a loop to repeatedly prompt the user until they do.

► Write a function which takes a `char` parameter and returns `true` if it is a numeric character (`0` through 9) and false otherwise.

► Using the function from the previous, write a function which takes a `string` parameter and returns true if every character in it is numeric (use a loop).

► What will the following program print as output?

```
int f(int x) {
    x *= 3;
    cout << x << endl;
    return x;
}

int g(int y) {
    cout << y - 1 << endl;
    return f(y) + 1;
}
```

```
int h(int x, int y) {
    cout << x + y << endl;
    x = 1;
    return x * y;
}

int main() {
    int x = 3, y = 5;
    cout << f(h(g(x), f(y))) << endl;
    return 0;
}
```

2 Vectors and Arrays

► Translate the vector variable declaration

```
vector<string> colors = {"red", "orange", "yellow", "green",
                         "blue", "indigo", "violet" };
```

into an array variable declaration.

► Given a vector `v`:

```
vector<int> v;
```

Draw the contents of the vector that will result after the following code is executed:

```
v.resize(5,10);
v.pop_back();
v.insert(v.begin() + 2, 13);
v.push_back(-1);
v.erase(v.begin() + 0);
v.push_back(-4);
```

► Write a function that will read in `float`s from the user until they press Ctrl-D and then return a vector containing every value entered.

► Write a function which takes a `vector<int>` parameter and which returns `true` if the vector contains any odd numbers, and false otherwise.

► Write a function which takes a `int n` parameter and which returns a vector containing the integers from 1 to n. E.g., if $n = 4$ then the vector returned should contain `{1,2,3,4}`. If the parameter is `0` or negative the returned vector should be empty.

► Suppose vectors did not have a `.size()` operation, only a `.empty()` operation (returns true if the vector is empty). Could you still write a function which determined the size (number of elements) in the vector? Write a substitute for the `.size()` operation:

```
int size(vector<int> v)
```

► What are the restrictions that arrays have, compared to vectors?

3 References, Pointers, and Dynamic Memory

► What will the values of the variables a, b, c be after the following code executes?

```
int a = 5, b = 6, c = 7;
int& d = b;
int& e = a;
a += b + d;
b *= c - e;
c -= a + b - d - e;
d *= 2;
e = a * b + c * d - e;
```

► What are the differences between references and pointers?

► What is wrong with the following code fragment:

```
int* p = nullptr;
{
    int x = 12;
    p = &x;
}
*p = 13;
```

► Use reference parameters to write a function `clamp`:

```
void clamp(int& x, int low, int high);
```

The effect should be to constrain the value of x to be in the range $[\texttt{low}, \texttt{high}]$. If $x < \texttt{low}$ then set x to `low`; if $x > \texttt{high}$ then set x to `high`, otherwise leave x unchanged.

- What is the *type* of the following variables?

```
int             x = 1;
int&            y = x;
int*            z = &y;
int*&           a = z;
int**           b = &a;
vector<int*>    vp;
vector<int*>&   vr = vp;
vector<int*>*   vpp = &vp;
vector<int*>** vppp = &vpp;
```

- What will be the final values of the variables a, b, c after the following code fragment is executed:

```
int a = 1, b = 2, c = 4;
int* p = &c;
int* q = &b;
int* r = &a;
*p = b;
*r = a;
p = r;
r = q;
q = &a;
*p *= *q;
*q += *r + a;
*r -= *p;
```

- What is the difference between `delete` and `delete[]`, and when is each used?

- Write a function

```
int* read_ints(int n);
```

which reads in exactly n integers from the user and then returns a pointer to a *dynamically allocated* array containing the values entered.

- Explain the difference between the *scope* of a variable and the *lifetime* of a value.