Lecture #5 | C/C++: array, C string, pointer

SE271 Object-oriented Programming (2017)

Prof. Min-gyu Cho

Today's Topic

- `main()` revisited
- Casting
- Array
- C string
- Pointer

`main()`

- Valid form of `main()` function in C++

```
int main() { /* ... */ }
int main(int argc, char* argv[]) { /* ... */ }
```

- If there is no explicit return statement for `main()`, the value returned is 0, meaning successful execution

Source: http://www.stroustrup.com/bs_faq2.html#void-main

Note: casting

- Casting: `(data_type)expression`
  - Forcefully coverts a value of the given expression into that of *data_type*
  - Frequently used in (old) C, but NOT recommended at all
- C++ provides `static_cast: static_cast<new_type>(expression)`
  - NOT recommended either
- Example

```
/* casting examples */
double d = 3.14;
int i1 = (int)d;                    // c-style casting
int i2 = static_cast<int>(d); // c++-style casting
```

Array

- Array: stores multiple elements of the same data type
- Declaration: *element_type array_name*[*constant_expression*]
  - Array index: `0, 1, ..., constat_expression - 1`
- Indexing: *array_name*[*index*]
- Examples:

```
int arr[3];
arr[0] = 2;
arr[1] = 3;
arr[2] = 5;
arr[3] = 7; // error?
```

Array: initialization

```
int a1[4];

int a2[4] = {1, 2, 3, 4};

int a3[4] = {1, };

int a4[4] = {};

int a5[] = {1, 2, 3, 4};

int a6[] {1, 2, 3, 4};
```

Array with loop

- Loop (for or while) is frequently used to access each element in array
- c.f., We will cover range-based for loop later (C++11)

```
int arr[4] {1, 2, 3, 4};
for (int i = 0; i < 3; i++)
    cout << "arr[" << i << "]=" << arr[i] << endl;
```

C string

- char: a character (depends on system/context) between single quotation marks (')
- C String: a null-terminated string between double quotation marks(")
  - An array of character
  - Ends with `'\0'`
- Will cover C++ standard string class later

Example: C string

```
int main(void)
{
    char s1[] = "Hello, world!";
    for (int i = 0; i < strlen(s1); i++)
        cout << "s1[" << i << "]=" << s1[i] << endl;
    cout << "s1[strlen(s1)]=" << s1[strlen(s1)] << endl;
}
```

```
s1[0]=H
s1[1]=e
s1[2]=l
s1[3]=l
s1[4]=o
s1[5]=,
s1[6]=
s1[7]=w
s1[8]=o
s1[9]=r
s1[10]=l
s1[11]=d
s1[12]=!
s1[strlen(s1)]=0
```

Pointer

- When we declare a variable, we associate the variable name with a particular location in memory (memory addresses or *pointers*), and stores a value there
- When we refer to a variable in C/C++, a computer take the following steps
  - Look up the address of the referred variable
  - Retreive or set the value in the above address
- Two operators
  - Address-of (&): evaluates the address of the given variable
  - Dereference (*): retrieves or sets the value at that location in memory

Example: pointer

```
int x = 42;
cout << "&x=" << &x << endl;
cout << "*(&x)=" << *&x << endl;

*(&x) = 23;
cout << "&x=" << &x << endl;
cout << "*(&x)=" << *&x << endl;
```

Pointer variable

- We can define a pointer variable, which takes a pointer (i.e., memory address) of variables of a specific data type
- Syntax: *data_type* pointer_variable;*

```
int x = 42;
int* ptr = &x;
cout << "ptr=" << ptr << endl;
cout << "*ptr=" << *ptr << endl;
*ptr = 6;
cout << "ptr=" << ptr << endl;
cout << "*ptr=" << *ptr << endl;
```

```
ptr=0x7fff5750782c
*ptr=42
ptr=0x7fff5750782c
*ptr=6
```

Arrays, pointers and pointer arithmetic

- Name of an array: a pointer to the first element in the array
- Pointer arithmetic
  - Using subtraction and addition of pointers to move around between locations in memory, typically between array element
  - Adding an integer *n* to a pointer produces a new pointer pointing to *n* positions further down in memory.
- Example

```
int i;
data_type arr;
data_type* ptr;
*(ptr + i) == arr[i]; // this is always true
```

Example: array and pointer

```
int a[] = {2, 3, 5, 7, 11, 13, 17};
int* ptr = a;
int array_size = sizeof(a) / sizeof(a[0]);

for (int i = 0; i < array_size; i++)
    cout << "ptr[" << i << "] = " << ptr[i] << endl;
```

```
ptr[0] = 2
ptr[1] = 3
ptr[2] = 5
ptr[3] = 7
ptr[4] = 11
ptr[5] = 13
ptr[6] = 17
```

Example: array and pointer (cont.)

```
// pointer arithmetic
cout << "ptr     = " << ptr << endl;
cout << "ptr + 1 = " << ptr + 1 << endl;
cout << "ptr + 2 = " << ptr + 2 << endl;
cout << "sizeof(*ptr) = " << sizeof(*ptr);
cout << endl;

for (int i = 0; i < array_size; i++) {
    cout << "*(ptr + " << i << ") = ”;
    cout << *(ptr + i) << endl;
}
ptr++;
cout << "ptr = " << ptr << endl;
cout << "*ptr = " << *ptr << endl;
```

```
ptr     = 0x7fff591c9870
ptr + 1 = 0x7fff591c9874
ptr + 2 = 0x7fff591c9878
sizeof(*ptr) = 4
*(ptr + 0) = 2
*(ptr + 1) = 3
*(ptr + 2) = 5
*(ptr + 3) = 7
*(ptr + 4) = 11
*(ptr + 5) = 13
*(ptr + 6) = 17
ptr = 0x7fff591c9874
*ptr = 3
```

Reading List

- Learn C++
  - Chapter 6: 1, 2, 3, 6, 7, 8, 8a
  - Chapter 7: 1, 2, 3, 4, 5, 8
- What good is `static_cast`?
  - http://www.stroustrup.com/bs_faq2.html#static-cast

ANY QUESTIONS?