Lab 3

Data Structures
C++ for C Coders

한동대학교 김영섭교수
idebtor@gmail.com

string, vector<string>
function overloading
new & delete operators
compilation of multiple files

Lab 3: string, and vector<string>

Step 1: Write two functions that take **`argc`** and **`argv`** as arguments and return the same information in their return type, respectively.

- **`string *args_to_strArray()`**
- **`vector<string> args_to_strVector()`**

Step 2: Write two functions that print its arguments.

- **`void print_args(vector<string> args)`**
- **`void print_args(string *args, int argc)`**

Step 3: Once you code them, move four functions developed above into a new file called **`args_to.cpp`**.

Step 4: To test your functions in multiple files, build an executable with two files, **`args.cpp`** and **`args_to.cpp`**.

```
// args.cpp
#include <iostream>
using namespace std;

int main(const int argc, char** argv) {
  cout << "You entered: "
       << argc << " arguments:" << endl;

  for (int i = 0; i < argc; ++i)
    cout << argv[i] << endl;
  return 0;
}
```

Lab 3 C++ class: string, and vector<string>

```
string *args_to_strArray(int argc, char **argv);
void print_args(int argc, string *strs);
```

```
vector<string> args_to_strVector(int argc, char **argv);
void print_args(vector<string> strs);
```

Using new & delete

- The **new** operator allocates memory, and **delete** frees it.

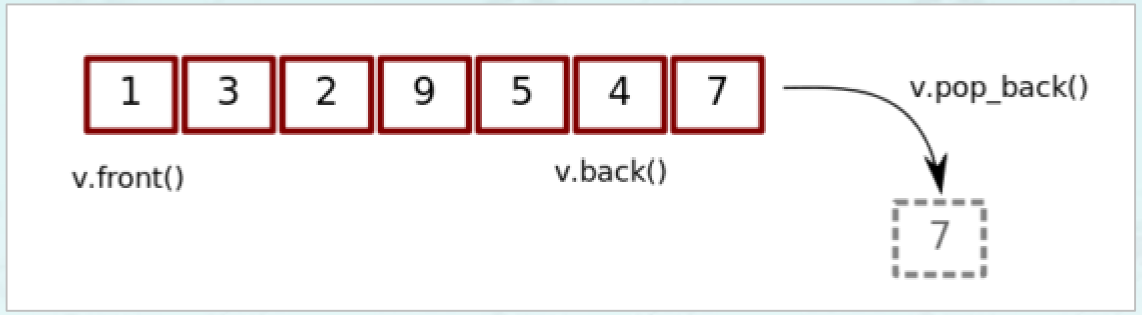
```
int *pi = new int;                     // pi points to uninitialized int
int *pi = new int(7);                  // which pi points has value 7
string *ps = new string("hello");   // ps points "hello", *ps = hello

int *pia = new int[7];                 // array of seven uninitialized ints
int *pia = new int[7]();               // array of seven ints values initialized to 0

string *psa = new string[5];                     // array of 5 empty strings
string *psa = new string[5]();                   // array of 5 empty strings
int    *pia = new int[5]{0, 1, 2, 3, 4};      // array of 5 ints initialized
string *psa = new string[2]{"a", "the"};      // array of 2 strings initialized
delete   pi;
delete[] pia;
```

Using vector and vector<string>

- In general, arrays are non-dynamic. It is static. That is to say, they are of fixed size. Vector in C++, however, allows us to store data in dynamic arrays.
- Vectors can resize itself automatically when an element is inserted or deleted depending on the need of the task to be executed.



```
#include <vector>
int main() {
  std::vector<int> v;
  v.push_back(1); v.push_back(3);
  ...
  std::cout << v.pop_back() << std::endl;
```

```
  for (int i = 0; i < v.size(); i++)
    std::cout << v[i] << std::endl;

  for (auto e = v.begin(); e != v.end(); e++)
    std::cout << *e << std::end;

  return 0
}
```

Multiple Source Files

- If you have multiple files to compile and link, for example,
  - Filename: **`args.cpp`**
  - Filename: **`args_to.cpp`**

- Compile and execute

```
$ g++ args.cpp args_to.cpp -o args
$ ./args
```

Lab 03:

- Files to submit:
  - **`args.cpp, args_to.cpp`**
- Due:
  - 11:55 pm March 12, 2020
- Grade:
  - 0.5

Lab 3

Data Structures
C++ for C Coders

한동대학교 김영섭교수
idebtor@gmail.com

string, vector<string>
function overloading
new & delete operators
compilation of multiple files