LAB II LECTURE 05
Modern C++

Seoul National University
Graphics & Media Lab

Mission

- Implement a program
  - Input
    - Arbitrary number of integers (using while(cin >> num) != EOF)
    - When “CTRL+d” is entered, finish input loop
  - Print out the numbers in descending order
  - Condition
    - Using STL::vector
    - Using auto keyword
    - Using range-based for loop

Contents

- What is modern C++?

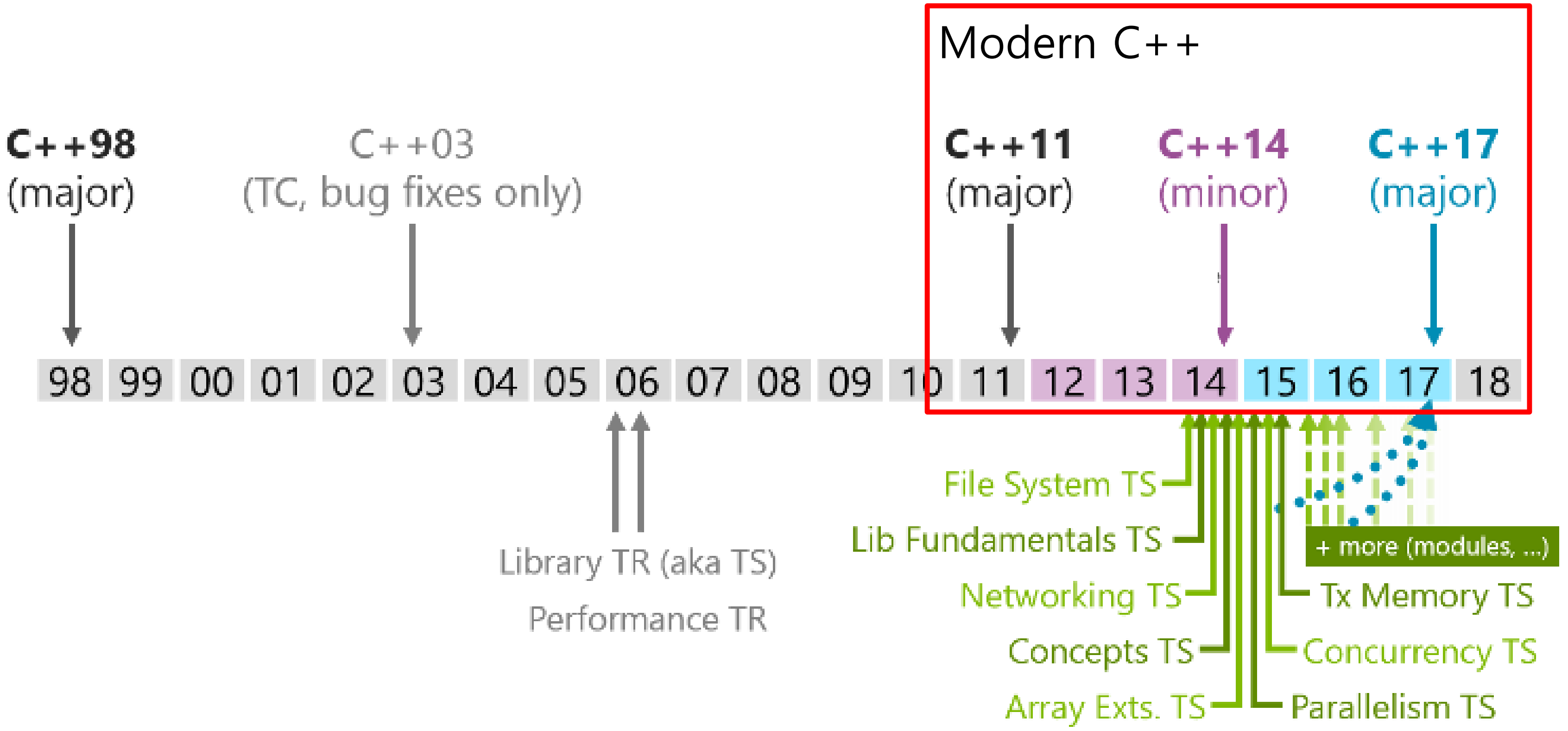
- New features in modern C++ which are covered in this class:
  - **nullptr**
  - **Automatic Type Deduction**
  - **Range based for loop**

History of C++

- Developed by Bjarne Stroustrup in 1985
  - Extension of the C language

- Standardized by ISO in 1998 (C++98)

C++ Standard


Modern C++
C++98 (major)
C++03 (TC, bug fixes only)
C++11 (major)
C++14 (minor)
C++17 (major)
98 99 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18
Library TR (aka TS)
Performance TR
File System TS
Lib Fundamentals TS
Networking TS
Concepts TS
Array Exts. TS
+ more (modules, ...)
Tx Memory TS
Concurrency TS
Parallelism TS

New Feature in Modern C++

- Lambda expression
- Automatic type deduction and ‘decltype’
- Uniform initialization syntax
- Deleted and defaulted functions
- nullptr
- Range-based for loop
- Strongly-typed enums
- Smart pointers

⋮

nullptr

- Use 0 or NULL when pointer is empty

```
#include <iostream>

void main(){
    char *cp = NULL;
    char *cp1 = 0;
}
```

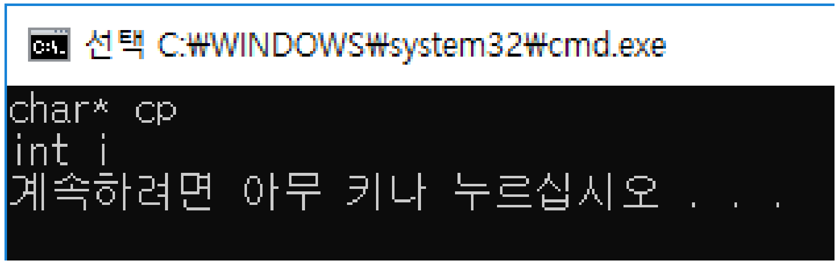
- In C++, type of NULL is ‘int’
  - When use NULL as parameter of function
    - Function call of ‘int’ type of parameter

```
#include <iostream>

using namespace std;

void f(char* cp){
    cout << "char* cp" << endl;
}

void f(int i){
    cout << "int i" << endl;
}

void main(){
    char *cp = NULL;
    f(cp);
    f(NULL);
    f(0);
}
```

nullptr

- Use nullptr instead of NULL or 0

```
#include <iostream>

using namespace std;

void f(char* cp){
    cout << "char* cp" << endl;
}

void f(int i){
    cout << "int i" << endl;
}

void main(){
    int *ip = nullptr;
    f(nullptr);
    f(0);
}
```



선택 C:₩WINDOWS₩system32₩cmd.exe

```
char* cp
int i
계속하려면 아무 키나 누르십시오 . . .
```

Automatic Type Deduction

- In C++, you must specify the type of an object when you declare it.
  - Static programming language
  - Java, C#...

C++

```
void main(){
    int a = 1;
    char c = 'a';
    double d = 0.1;
}
```

- In dynamic programming language
  - Type of variable is automatically deduced as the program is compiled.
  - Python, JavaScript ...

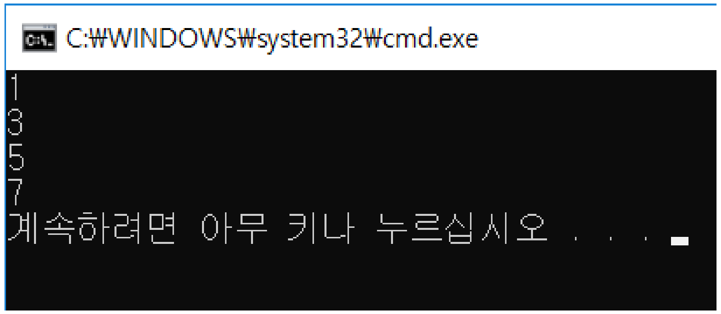
Python

```
x = 34 - 23
y = "Hello"
z = 3.45
```

Python

```
x = 34 - 23
print(x)
x = "Hello"
print(x)
```

Modern C++ Allows Automatic Type Deduction



```
C:₩WINDOWS₩system32₩cmd.exe
1
3
5
7
계속하려면 아무 키나 누르십시오 . . .
```

- When you use template or more advanced feature, defining type of return variable is difficult

```cpp
#include <iostream>
#include <vector>

void main(){
    std::vector<int> vec;
    vec.push_back(1);
    vec.push_back(3);
    vec.push_back(5);
    vec.push_back(7);

    for(std::vector<int>::iterator it = vec.begin(); it<vec.end(); it++){
        std::cout << *it << std::endl;
    }
}
```

- Use keyword ‘auto’

```cpp
for(auto it = vec.begin(); it<vec.end(); it++){
    std::cout << *it << std::endl;
}
```

Modern C++ Allows Automatic Type Deduction

- Keyword ‘auto’
  - Compiler infers the type of the variable
    - Assignment
    - Return type from function

```
void main(){
    auto d = 5.0;
    auto i = 1+2;
}
```

```
int add(int x, int y){
    return x+y;
}

void main(){
    auto sum = add(5,6);
}
```

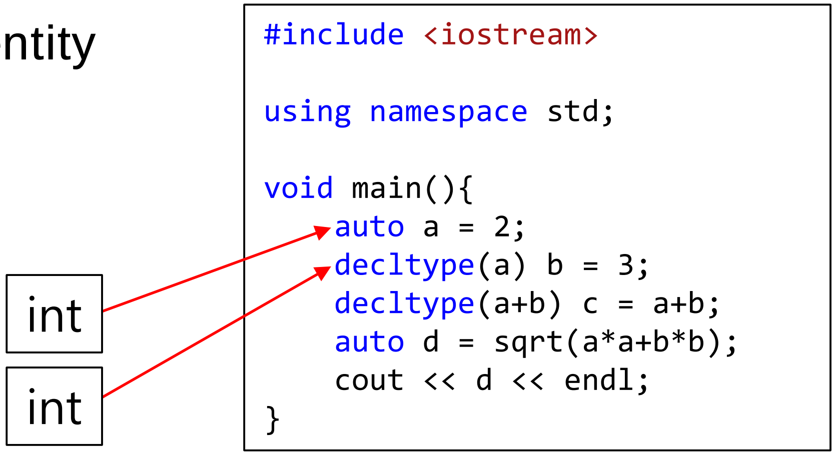
- Can not use
  - Declare without initialization
  - Parameter of function

```
auto d = 5.0; // OK
auto a;       // ERROR
```

```
int f(auto i, int x){
    return x+i;
}
```

Modern C++ Allows Automatic Type Deduction

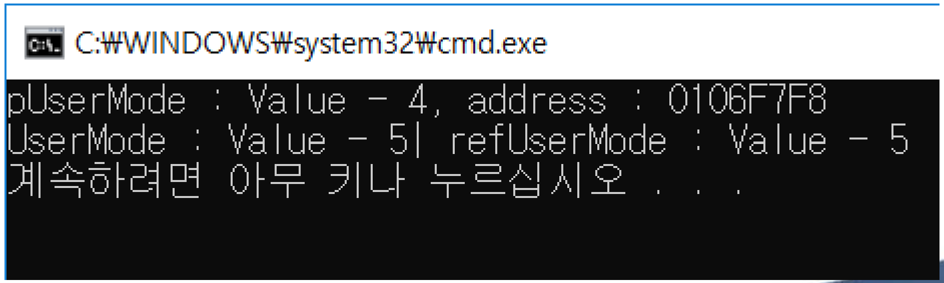
- Keyword ‘decltype’
  - decltype(entity)
  - Inspects the declared type of an entity



```
#include <iostream>

using namespace std;

void main(){
    auto a = 2;
    decltype(a) b = 3;
    decltype(a+b) c = a+b;
    auto d = sqrt(a*a+b*b);
    cout << d << endl;
}
```

Practice

```cpp
#include <iostream>

using namespace std;

void main(){
    int UserMode = 4;
    auto* pUserMode = &UserMode;
    cout << "pUserMode : Value - " << *pUserMode
    << ", address : " << pUserMode << std::endl;

    decltype(UserMode)& refUserMode = UserMode;
    refUserMode = 5;
    cout << "UserMode : Value - " << UserMode
    << "| refUserMode : Value - " << refUserMode << endl;
}
```


C:₩WINDOWS₩system32₩cmd.exe

```
pUserMode : Value - 4, address : 0106F7F8
UserMode : Value - 5| refUserMode : Value - 5
계속하려면 아무 키나 누르십시오 . . .
```

Range Based For Loop

- C++ for loop
  - for(initialization, condition, increment or decrement)

```
#include <iostream>

using namespace std;

void main(){
    int numList[5] = {1,2,3,4,5};

    for(int i=0; i<5; i++){
        cout << numList[i] << endl;
    }
}
```

- Range based for Loop
  - for(declaration, expression)

```
void main(){
    int numList[5] = {1,2,3,4,5};

    for(auto i : numList){
        cout << i << endl;
    }
}
```

Range Based For Loop

```
void main(){
    int numList[5] = {1,2,3,4,5};

    for(auto i : numList){
        cout << i << endl;
    }
}
```

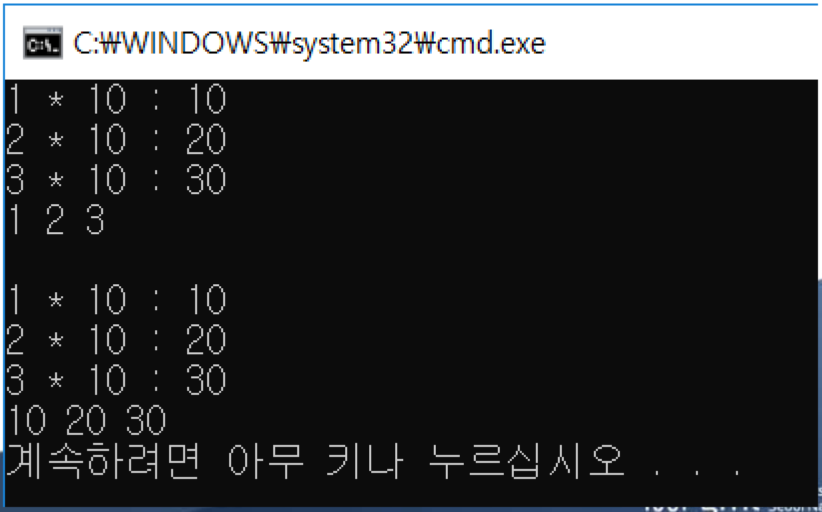
- Executes a for loop over a range

- Limitation
  - Can not customize
    - Print only first 3 elements in array of size 5
  - Forward loop

Practice

```
#include <iostream>
#include <vector>

using namespace std;

void main(){
    vector<int> NumberList;
    NumberList.push_back(1);
    NumberList.push_back(2);
    NumberList.push_back(3);

    for(auto i : NumberList){
        cout << i << " * 10 : ";
        i *= 10;
        cout << i << endl;
    }

    for(auto i : NumberList)
        cout << i << " ";
```

```
    cout << endl << endl;

    for(auto &i : NumberList){
        cout << i << " * 10 : ";
        i *= 10;
        cout << i << endl;
    }

    for(auto i : NumberList)
        cout << i << " ";

    cout << endl;
}
```


C:₩WINDOWS₩system32₩cmd.exe

```
1 * 10 : 10
2 * 10 : 20
3 * 10 : 30
1 2 3

1 * 10 : 10
2 * 10 : 20
3 * 10 : 30
10 20 30
계속하려면 아무 키나 누르십시오 . . .
```

Mission

- Implement a program
  - Input
    - Arbitrary number of integers (using while(cin >> num))
    - When “CTRL+d” is entered, finish input loop
  - Print out the numbers in descending order
  - Condition
    - Using STL::vector
    - Using auto keyword
    - Using range-based for loop