ECE4820J Lab 6 Topic 1

> Author: 🐹 + 🚗

auto

`auto` is a C++ key word, two main usage:

- Auto deduct variable type at declaration
- Use as function return type

Example 1

```
auto f = 3.14;  //double
auto s("hello");  //const char*
auto z = new auto(9);  //int *
auto x1 = 5, x2 = 5.0, x3 = 'r';   //wrong, must be same type
```

It's not suggested to use `auto` for those easy types

Example 2

```
 std::vector<int> vect;
 for(auto it = vect.begin(); it != vect.end(); ++it)
   //the type of "it" is std::vector<int>::iterator
 for(auto &item : v) cout << item << endl;
   // list for
   // support from C++11, often used in leetcode
```

Example 3

```
unsigned int a=4294967295;
unsigned int b=1;
auto c=a+b;  // still unsigned int, overflow
```

Example 4

Can be used with & and *.

```
int* p = new auto(0); //fine
int** pp = new auto(); // wrong
auto x = new auto(); // wrong
auto* y = new auto(9); // Fine. Here y is a int*
auto z = new auto(9); //Fine. Here z is a int* (It is not just an int)
const auto a = 6;
```

Example 5

Can be used with lambda function, no need to include `<functional>`.

```
auto ptr = [](double x)
    {return x*x;};  //type: std::function<double(double)>
```

decltype

`decltype` checks the data type of a variable/expression, and uses it to declare another variable.

Example 1

```
int a = 1;

decltype(a) b = 2; // b is int
decltype(a+1) c = 2; // a+1 is int so c is int. Compiler does not actually
calculate a+1.
```

Example 2

```
int i = 1;
int *p = &i;
int &r = i;

decltype(p) b; // b is int*
decltype(r+0) c; // r+0 is an expression. It returns a rvalue as a int. So c is
int.
decltype(*p) d; // *p is an expression rather than a variable. *p can be used as
a lvalue, so decltype returns a reference to the lvalue. Then d is int&.
decltype((i)) e; // If you add a '()' to the variable, compiler treats it as an
expression. Since it can become a lvalue, e is int&.
```

Example 3

```
const int &r = 1;
auto a = r; //auto loses all the modifiers. a is int.
decltype(r) b = 1; //decltype keeps the modifiers. b is const int &.
decltype(auto) c = r; //decltype(auto) keeps all the modifiers of r, so c is
exaclty the same as r (const int &)
```

Example 4

```
int f(){return 1;}
int g(){return 2;}

decltype(f()) d = 2; // Function f returns an int. d is int.
decltype(f) e; // e is int(void). Can be called and used as a function template.

void ss(decltype(f) e){std::cout <<e();}
ss(g); // 2
```

Example 5

```
//trailing-return-type
//Used when we don't care the return type of a function

//WRONG:
template <typename T, typename U>
decltype(t+u) add(T t,U u) //Compile Error. When processing "decltype(t+u)", t and u have not been declared!
{
    return t+u;
}

//CORRECT:
template <typename T, typename U>
auto add(T t,U u) ->decltype(t+u) //Return type of function add is the same as the type of (t+u)
{
    return t+u;
}
```

Note: `auto` and `decltype` are introduced in C++11.