601.220 Intermediate Programming

Template classes

Template classes

When we saw STL, we considered these templates:

```
template<typename T>
struct Node {
    T payload;
    Node *next;
};

template<typename T>
void print_list(Node<T> *head) {
    Node<T> *cur = head;
    while(cur != NULL) {
        cout << cur->payload << " ";
        cur = cur->next;
    }
    cout << endl;
}
```

One `struct`/function that works for (almost) any type `T`

Template classes

```
int main() {
    Node<float> f3 = {95.1f, NULL}; // float payload
    Node<float> f2 = {48.7f, &f3};  // float payload
    Node<float> f1 = {24.3f, &f2};  // float payload
    print_list(&f1);

    Node<int> i2 = {239, NULL};     // int payload
    Node<int> i1 = {114, &i2};      // int payload
    print_list(&i1);

    return 0;
}
```

Template classes

```
// ll_template_cpp.cpp:
#include <iostream>

using std::cout;   using std::endl;

template<typename T>
struct Node {
    T payload;
    Node *next;
};

template<typename T>
void print_list(Node<T> *head) {
    Node<T> *cur = head;
    while(cur != NULL) {
        cout << cur->payload << " ";
        cur = cur->next;
    }
    cout << endl;
}

int main() {
    Node<float> f3 = {95.1f, NULL};
    Node<float> f2 = {48.7f, &f3};
    Node<float> f1 = {24.3f, &f2};
    print_list(&f1);

    Node<int> i2 = {239, NULL};
    Node<int> i1 = {114, &i2};
    print_list(&i1);

    return 0;
}
```

Template classes

```
$ g++ -c ll_template_cpp.cpp -std=c++11 -pedantic -Wall -Wextra
$ g++ -o ll_template_cpp ll_template_cpp.o
$ ./ll_template_cpp
24.3 48.7 95.1
114 239
```

Now we make it a template class.

Template classes

```
// ll_temp.h:
#include <iostream>

template<typename T>
class Node {
public:
    Node(T pay, Node<T> *nx) : payload(pay), next(nx) { }

    void print() const {
        const Node<T> *cur = this;
        while(cur != NULL) {
            std::cout << cur->payload << ' ';
            cur = cur->next;
        }
        std::cout << std::endl;
    }

private:
    T payload;
    Node<T> *next;
};
```

Template classes

```
// ll_temp_main.cpp:
#include "ll_temp.h"

int main() {
    Node<float> f3 = {95.1f, NULL};
    Node<float> f2 = {48.7f, &f3};
    Node<float> f1 = {24.3f, &f2};
    f1.print();

    Node<int> i2 = {239, NULL};
    Node<int> i1 = {114, &i2};
    i1.print();

    return 0;
}
```

```
$ g++ -o ll_temp_main ll_temp_main.cpp -std=c++11 -pedantic -Wal
$ ./ll_temp_main
24.3 48.7 95.1
114 239
```

Template classes

Everything looks fine. Now lets separate the template class into a `.h` file and a `.cpp` file.

```
// ll_temp2.h:
// ll_temp2.h
#include <iostream>

template<typename T>
class Node {
public:
    Node(T pay, Node<T> *nx) : payload(pay), next(nx) { }
    void print() const;
private:
    T payload;
    Node<T> *next;
};
```

Template instantiation

```
// ll_temp2.cpp:
// ll_temp2.cpp
#include "ll_temp2.h"

template<typename T>
void Node<T>::print() const {
    const Node<T> *cur = this;
    while(cur != NULL) {
        std::cout << cur->payload << ' ';
        cur = cur->next;
    }
    std::cout << std::endl;
}
```

Template instantiation

```
// ll_temp_main2.cpp:
#include "ll_temp2.h"

int main() {
    Node<float> f3 = {95.1f, NULL}; // instantiate Node<float>
    Node<float> f2 = {48.7f, &f3};
    Node<float> f1 = {24.3f, &f2};
    f1.print(); // instantiate Node<float>::print  *** will this work? ***

    Node<int> i2 = {239, NULL};    // instantiate Node<float>
    Node<int> i1 = {114, &i2};
    i1.print(); // instantiate Node<int>::print    *** will this work? ***

    return 0;
}
```

int

```
$ g++ ll_temp_main2.cpp ll_temp2.cpp -std=c++11 -pedantic -Wall -Wextra
/usr/bin/ld: /tmp/ccuI6P9d.o: in function `main':
ll_temp_main2.cpp:(.text+0x72): undefined reference to `Node<float>::print() co
/usr/bin/ld: ll_temp_main2.cpp:(.text+0xa9): undefined reference to `Node<int>:
collect2: error: ld returned 1 exit status
```

Template instantiation

Compiler acts *lazily* when instantiating templates

Doesn't instantiate until *first use*

```
Node<float> f3 = {95.1f, NULL}; // OK fine, I'll instantiate Node<float>
Node<float> f2 = {48.7f, &f3};
Node<float> f1 = {24.3f, &f2};
f1.print();                     // OK fine, I'll instantiate Node<float>::print

Node<int> i2 = {239, NULL};          // OK fine, I'll instantiate Node<int>
Node<int> i1 = {114, &i2};
i1.print();                      // OK fine, I'll instantiate Node<int>::print
```

Template instantiation

When instantiating, compiler needs the relevant template classes and functions *to be fully defined already*. . .

. . . *in contrast to typical function or class use*, where definition can be in separate `.cpp` files. i.e. template classes have to be typically defined in the header files.

Template instantiation

```
$ g++ ll_temp_main2.cpp ll_temp2.cpp -std=c++11 -pedantic -Wall -Wextra
/usr/bin/ld: /tmp/ccOGHNyP.o: in function `main':
ll_temp_main2.cpp:(.text+0x72): undefined reference to `Node<float>::print() co
/usr/bin/ld: ll_temp_main2.cpp:(.text+0xa9): undefined reference to `Node<int>:
collect2: error: ld returned 1 exit status
```

Lazily instantating `Node<float>` & `Node<int>` is fine

- Both are defined in the header

Instantating `Node<float>::print()` & `Node<int>::print()` *won't work*

- They are in a separate source file not `#included` here
- By convention, we never `#include` `.cpp` files

Template instantiation

There are a couple of possible solutions:

- Move `Node<T>::print()` definition back into the header
- Put member function definitions in a separate file and `#include` it, but *don't* give it a `.cpp` extension. i.e. a different extension such as `.inc` or `.inl`.

Template instantiation

```
// ll_temp2.inc:
// ll_temp2.inc
template<typename T>
void Node<T>::print() const {
    const Node<T> *cur = this;
    while(cur != NULL) {
        std::cout << cur->payload << ' ';
        cur = cur->next;
    }
    std::cout << std::endl;
}
```

Template instantiation

```
// ll_temp_main3.cpp:
#include "ll_temp2.h"   // template class definition
#include "ll_temp2.inc" // template class member function definitions

int main() {
    Node<float> f3 = {95.1f, NULL}; // instantiate Node<float>
    Node<float> f2 = {48.7f, &f3};
    Node<float> f1 = {24.3f, &f2};
    f1.print(); // instantiate Node<float>::print  *** will this work? ***

    Node<int> i2 = {239, NULL};    // instantiate Node<float>
    Node<int> i1 = {114, &i2};
    i1.print(); // instantiate Node<int>::print    *** will this work? ***

    return 0;
}
```

```
$ g++ -o ll_temp_main3 ll_temp_main3.cpp -std=c++11 -pedantic -W
$ ./ll_temp_main3
24.3 48.7 95.1
114 239
```