Week 3

Exercise 19

../19/binary.h

```
1  #ifndef INCLUDED_BINARY_
2  #define INCLUDED_BINARY_
3
4  #include <cstddef>
5
6  template <size_t n0>
7  struct Bin
8  {
9    enum
10   {
11     value = Bin<(n0 >> 1)>::value * 10 + (n0 & 1)
12   };
13 };
14
15 template <>
16 struct Bin<0>
17 {
18   enum
19   {
20     value = 0
21   };
22 };
23
24
25 #endif
```

../19/main.ih

```
1  #define ERR(msg) printf("%s : %d", (msg), __LINE__)
2
3  #include "binary.h"
4
5  #include <iostream>
6
7  using namespace std;
```

../19/main.cc

```
1  #include "main.ih"
2
3  int main()
4  {
5      cout << Bin<5>::value  << '\n'
6           << Bin<27>::value << '\n';
7  }
```

Exercise 20

../20–2/main.ih

```
#define ERR(msg) printf("%s : %d", (msg), __LINE__)

#include "chars/chars.h"
#include "onechar/onechar.h"
#include "merge/merge.h"

#include <iostream>

using namespace std;
```

../20–2/main.cc

```
#include "main.ih"

int main(int argc, char const **argv)
{
  cout <<
    Merge<
        Chars<'1', '2', '3'>,
        Merge<Chars<'4', '5'>, OneChar<'6'>>::CP
      >::CP() << '\n';
}
```

../20–2/chars/chars.h

```
#ifndef INCLUDED_CHARS_
#define INCLUDED_CHARS_

#include <iostream>

template <char ...CharsT>
class Chars
{
  public:

  constexpr static std::string letters();

  template <char... CharsT2>
  friend std::ostream &operator<<(std::ostream &out,
                                  Chars<CharsT2...> const &rhs);
};


template <char... CharsT>
constexpr std::string Chars<CharsT...>::letters()
{
  return std::string{ CharsT... };
}


template <char... CharsT>
std::ostream &operator<<(std::ostream &out,
                                Chars<CharsT...> const &rhs)
{
  out << rhs.letters();
  return out;
}



#endif
```

../20–2/merge/merge.h

```
1  #ifndef INCLUDED_MERGE_
2  #define INCLUDED_MERGE_
3
4  template <class CharsT1, class CharsT2>
5  class Merge
6  {};
7
8  template <char ...CharsT1, char CharT2>
9  class Merge <Chars<CharsT1...>, OneChar<CharT2>>
10 {
11   public:
12
13     typedef Chars<CharsT1..., CharT2> CP;
14
15 };
16
17 template <char ...CharsT1, char ...CharsT2>
18 class Merge <Chars<CharsT1...>, Chars<CharsT2...>>
19 {
20   public:
21
22     typedef Chars<CharsT1..., CharsT2...> CP;
23 };
24
25
26
27 #endif
```

../20–2/onechar/onechar.h

```
1  #ifndef INCLUDED_ONECHAR_
2  #define INCLUDED_ONECHAR_
3
4  template<char character>
5  class OneChar
6  {};
7
8  #endif
```

Exercise 21

../21/type/type.h

```
#ifndef INCLUDED_TYPE_
#define INCLUDED_TYPE_

#include "../typeidx/typeidx.h"

template<typename NeedleT, typename ...HayStackT>
class Type
{
  public:

    enum
    { //TypeIdx starts looking at position 1 in the haystack
      located = 0 + TypeIdx<1, NeedleT, HayStackT...>::located
    };

  private:
};

template<typename NeedleT> //if there is only a needle eg Type<int>::located
class Type<NeedleT>
{
  public:
    enum
    {
      located = 0
    };
};




#endif
```

../21/type/type.ih

```
#include "type.h"

using namespace std;
```

../21/typeidx/typeidx.h

```
#ifndef INCLUDED_TYPEIDX_
#define INCLUDED_TYPEIDX_

template<int counter, typename NeedleT, typename nextT, typename ...RestT>
class TypeIdx
{
  public:

    enum //nothing is added from this point if sizes(and thus types) are equal
    {    //in which case located is equal to the counter
      located = (sizeof(NeedleT) == sizeof(nextT)) * (counter) +
                (sizeof(NeedleT) != sizeof(nextT)) *
                            TypeIdx<counter + 1, NeedleT, RestT...>::located
    }; //calls itself until only one parameter is left in RestT which then
};     //calls the corresponding specialization, ending the recursion

template<int counter, typename NeedleT, typename LastT>
class TypeIdx<counter, NeedleT, LastT>
{
  public:
```

```
21
22      enum //zero if not same type, otherwise equal to counter
23      {
24        located = (sizeof(NeedleT) == sizeof(LastT)) * (counter)
25      };
26  };
27
28  #endif
```

../21/typeidx/typeidx.ih

```
1  #include "typeidx.h"
2
3  using namespace std;
```

../21/main.ih

```
1  #define ERR(msg) printf("%s : %d", (msg), __LINE__)
2
3  #include "type/type.h"
4  #include "typeidx/typeidx.h"
5
6  #include <iostream>
7
8  using namespace std;
```

../21/main.cc

```
1  #include "main.ih"
2
3  int main()
4  {
5      cout <<
6          Type<int>::located << ' ' <<
7          Type<int, double>::located << ' ' <<
8          Type<int, int>::located << ' ' <<
9          Type<int, double, int>::located << ' ' <<
10         Type<int, double, int>::located << ' ' <<
11         Type<int, double, int, int, int>::located <<
12         '\n';
13 }
```