Семинар 1.10 «Вариативные шаблоны»

Зачем они нужны?

- [], std::array ~ std::pair - фиксированный
  new, std::vector [?]* - динамический

  гомогенные контейнеры — гетерогенные контейнеры => рассматриваем std::tuple (гетерогенность в compile-time)

  * гетерогенный run-time контейнер : std::vector + std::any

- Обертка вокруг другого объекта

```
constructor
  T* ptr =
    new T(...)
```

T - любой (неизвестный тип), обёртка - шаблон
Как написать к-р, передающий аргументы T?
Пример: std::thread и функция

- Функция, обрабатывающая неизвестное кол-во аргументов произвольных типов, например, print (см. ... и макросы C)

Пример (live) функция print (рекурсивное инстанцирование)

Перегрузка вариативных шаблонов

Пример (live) предпочитается версия без пакета параметров
sizeof... (args/Types) - ошибочное использование в if (...)

Выражения свёртки

```
template < typename ... T>
auto sum (T... s)
{ return (... + s); -> ((s1+s2) + s3) +... (бинарный оператор)
```

(... op P) -> (init op ... op P) -> ...
(P op ...) -> (P op ... op init) -> ...

Пример (live) print с выражением свёртки
Пример (live) print с выражением свёртки + пробел
Пример tree_traverse.cpp - обход дерева

Дополнительные возможности использования пакетов:

- Вариативные выражения

```
print (args + args ...); // 1... - ошибка
```

```
template < typename T1, typename ... TN >
[constexpr] bool is_homogeneous (T1, TN...) -> decay
{
  return (std::is_same_v<T1, TN> && ...);
}
```

true / false - одинаковые типы?

□ **ВАРИАТИВНЫЕ ИНДЕКСЫ**

```
template < typename C, typename ... Idx >
void print_by_idx (const C & c, Idx ... idx)
{
  print (c[idx]...);
}
template < typename C, std::size_t ... Idx >
void print_by_idx (const C & c)
{
  print (c[Idx]...);
}
```

Реализация статического полиморфизма

Динамический полиморфизм - наследование + в.ф. + * или &

ABC Shape + наследники Circle, Rectangle, Triangle и др.

```
|| virtual void draw () const = 0; // -> override
```

```
void draw (const Shape & s)
{
  s.draw();
}
```

```
template < typename S >
void draw (const S & s)
{
  s.draw();
}
```

базовый класс	VS	параметр шаблона
таблица в.ф.		инстанцирование
↓		↓
полный интерфейс		гастигный интерфейс
меньше размер кода		более быстрый код

Примечание: виртуальная функция не может быть шаблонной

CRTP - передача класса-потомка параметром шаблона родителя

```
template < typename D > - избавление от в.ф.
class Base {...};

class Derived : public Base < Derived >
```

Пример counter.cpp - счётчик экземпляров для разных классов