Subject: Re: get_i
Posted by mirek on Tue, 16 Jun 2020 22:01:23 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 16 June 2020 21:20Novo wrote on Tue, 16 June 2020 18:21Another experiment/suggestion.
I rewrote get_i using variadic template:

```
template <typename A, typename... T>
constexpr A get_i(int i, const A& p0, const T& ...args)
{
 A list[] = {p0, args...};
 int n = sizeof...(args);
 return list[clamp(i, 0, n)];
}

const char* cr = get_i(1, "zero", "one", "two");
RDUMP(cr);
int ir = get_i(1, 0, 1, 2);
RDUMP(ir);
```

IMHO, my implementation is much shorter and it will compile faster.
IMHO, macroses __List and __Expand are not needed anymore ...

Yes, you are right about this, I have used old tricks mostly out of habit. I guess I will have to rewrite it all now

Do you see any problems?

```
template <class T, class V>
constexpr V decode(const T& sel, const V& def)
{
 return def;
}

template <class T>
constexpr const char *decode(const T& sel, const char *def)
{
 return def;
}

template <class T, class K, class V, typename... L>
constexpr V decode(const T& sel, const K& k, const V& v, const L& ...args)
{
 return sel == k ? v : (V)decode(sel, args...);
}
```

```
template <class T, class K, typename... L>
constexpr const char *decode(const T& sel, const K& k, const char *v, const L& ...args)
{
 return sel == k ? v : decode(sel, args...);
}


template <class T, class K>
constexpr int findarg(const T& x, const K& k)
{
 return x == k ? 0 : -1;
}

template <class T, class K, typename... L>
constexpr int findarg(const T& sel, const K& k, const L& ...args)
{
 if(sel == k)
  return 0;
 int q = findarg(sel, args...);
 return q >= 0 ? q + 1 : -1;
}
```
