(Średnio) zaawansowane programowanie w C++ (ZPR)

Wykład 14 - LLM w wytwarzaniu oprogramowania. Przetwarzanie tekstu. Powtórzenie

Robert Nowak

25L

Zintegrowane narzędzia do tworzenia oprogramowania (IDE)[2]

Nazwa	Licencja	Popularność[1]	Platforma
VS Code	Komer. + wolna	73.6%	wiele
Visual Studio	Komer.	29.3%	Windows, macOS
JetBrains	Komer. + Apache 2.0	94.0%	wiele
Vim/Neovim	Vim + Apache 2.0	34.1%	wiele
Sublime Text	Komer. + wolna	10.9%	wiele
Eclipse	EPL 2.0	9.4%	wiele
XCode	Komer. + wolna	9.3%	macOS
VSCodium	MIT	4.8%	wiele
Emacs	GPL v3.0	4.6%	wiele

[1] Ankieta, 2024, 58tys. uczestników

[2] Na podstawie raportu M. Borek, 2024

Narzędzia wspierające tworzenie kodu[3]

Nazwa	Open Source	Objaśnianie	CLI	Praca lokalna
GitHub/Copilot	-	tak	tak	-
Tabnine	-	tak	-	tak
Codeium	-	tak	-	-
Amazon Whisperer	-	tak	tak	-
Cody	tak	tak	tak	-
Continue	tak	tak	-	tak
CodeGeeX	tak	tak	-	-
FauxPilot	tak	-	-	tak
Tabby	tak	tak	-	tak

[3] Na podstawie raportu M. Borek, 2024

Integracja narzędzi z IDE[4]

Nazwa	VS Code	V S	Vim/ Neo	Jet Bra.	Ecli- pse	Sub- lime	X Code	Cod- ium	Em- acs
Copilot	tak	tak	tak	tak	-	-	-	tak	tak
Tabnine	tak	tak	tak	tak	tak	-	-	tak	tak
Codeium	tak	tak	tak	tak	tak	tak	tak	tak	tak
Am. W.	tak	tak	-	-	-	-	-	tak	-
Cody	tak	tak	-	tak	tak	-	-	tak	-
Continue	tak	tak	-	tak	-	-	-	tak	-
CodeGee	tak	tak	-	tak	-	-	-	tak	-
FauxPilot	-	tak	tak	-	-	-	-	tak	-
Tabby	tak	tak	tak	tak	-	-	-	tak	-

[4] Na podstawie raportu M. Borek, 2024

Przetwarzanie tekstu w C++

przetwarzanie tekstu : konwersje za pomocą strumieni

```
//C++11 <string> definiuje konwersję dla wszystkich typów wbudowanych
string to_string(int val);
string to_string(double val);

//konwersja za pomocą strumieni, obsługuje typy wbudowane i użytkownika
#include <sstream>
std::ostringstream os;
os << 1234;
os.str();//Zwraca std::string

std::string napis("1234");
std::istringstream is(napis);
int i;
is >> i;

//ale ma niewygodny interfejs (potrzebne 3 linie kodu)
```

Boost.lexical_cast

```
  //Szablon wykorzystuje strumienie do przekształceń
template<typename Target, typename Source>
Target lexical_cast(const Source & arg) {
  std::stringstream sout;
  Target result;
  if(!(sout << arg && sout >> result) )
    throw bad_lexical_cast( typeid(Source), typeid(Target) );
  return result;
}
```

- 'Source' może być zapisywane do strumienia
- 'Target' może być czytany ze strumienia
- 'Source' i 'Target' mają konstruktory kopiujące
- 'Target' ma konstruktor bezparametrowy

wykorzystanie boost::lexical_cast

```
#include <boost/lexical_cast.hpp>
//Przekształcenie napisu na inny typ
int i = lexical_cast<int>(string("1234") );
//Przekształcenie typu na napis
string s = lexical_cast<string>(1234);
Foo f;
string s = lexical_cast<string>(f); //podobnie dla typu użytkownika
```

- obsługuje typy wbudowane i typy użytkownika,
- wygodny interfejs, obsługa błędów,
- zgłoszona do umieszczenia w bibliotece standardowej (TR2), ale nie została umieszczona w C++11, C++14, C++17, C++20

lokalizm (C++)

klasyfikacja znaków, porządek napisów, format daty, itp.

```
#include <locale>
class ios_base {
  locale imbue(const locale& loc);//ustawia lokalizm
  locale getloc() const; //pobiera lokalizm
};
```

- locale::classic() standardowy US English ASCII
- globalna
  - locale::locale() konstr. domyślny tworzy taką jak globalna
  - inicjowana na classic
  - zmiana na inną: locale::global()
- locale::locale(""); tworzy lokalizm domyślny dla systemu
- locale::locale("nazwa"); tworzy lokalizm o podanej nazwie. Może zgłosić wyjątek 'runtime_error'

Wsparcie dla Unicode

Dostępne w standardzie C++03

```
"ASCII String"//Tablica obiektów typu char
L"wchar_t String"//Tablica obiektów wchar_t
```

Dostępne w standardzie C++11

```
u8"UTF-8 String \u2018."//UTF-8, tablica obiektów typu char
u"UTF-16 String \u2018."//UTF-16, tablica obiektów char16_t
U"UTF-32 String \u2018."//UTF-32, tablica obiektów char32_t
```

Napisy ze znakami specjalnymi (raw string literals)

```
auto str = R"({
    "Title":"C/C++",
    "Id":6
})";
```

Własne symbole ogranicznika, R"ddd(...)ddd"

```
auto str = R"ZZZ({"Title":"(C/C++)","Id":6})ZZZ";
```

Wyrażenia regularne

- element programowania deklaratywnego
- przydatne przy przetwarzaniu tekstów
- dostępne standardowo w wielu językach programowania

```
template <class charT, class traits = regex_traits<charT> >
class basic_regex { //reprezentuje wyrażenie regularne
  //
};
using regex = basic_regex<char>;
using wregex = basic_regex<wchar_t>;
```

```
#include <regex>
std::regex reg("a(a|b)*b"); //obiekt repr. wyrażenie regularne
#include <boost/regex.hpp>
boost::regex reg2("a(a|b)*b"); //obiekt repr. wyrażenie regularne
```

opis wyrażenia regularnego `regex`

^ $	początek i koniec linii	
.	dowolny pojedynczy znak	.la 1la ala bla cla dla
[aeo]	zbiór znaków	[aeo]la ala ela ola
[a-z]	zakres znaków	
[^0-9]	znak spoza zakresu	
\d	klasy znaków [[:digit:]]	\dla 1la 2la 3la
\s	biały znak [[:space:]]	
*	dowolna liczba (zero lub więcej)	a*b b ab aaaab
+	jedno lub więcej	a+b ab aaaab
?	opcjonalność: zero lub jedno	a?b b ab
{m,n}	od m do n wystąpień	
(wyr)	tworzy grupę	(ab)+ ab abab ababab
|	alternatywa	a(a|b)b aab abb

Badanie, czy napis jest opisywany wyrażeniem regularnym

```
template </* ... */>
bool regex_match( //funkcja bada, czy napis jest opisany wyrażeniem
  const basic_string</* ... */>& str,
  const basic_regex </* ... */>& e,
  match_flag_type flags = match_default);

//Przykład użycia - badanie czy napis jest typu NNN-NNN-NN-NN
bool poprawny_NIP(const std::string& nip_str) {
  static const boost::regex e("\\d{3}-\\d{3}-\\d{2}-\\d{2}");
  return regex_match(nip_str, e);
}
```

Typowe wykorzystanie wyrażeń regularnych

```
regex reg("a*b");
//regex_search poszukuje napisu spełniającego wyrażenie
regex_search("aba",reg); //Zwróci true
//regex_match bada, czy napis spełnia wyr. regularne
regex_match("aba",reg); //Zwróci false
//regex_iterator iteracja po odnalezionych napisach
//regex_replace zastępuje napisy opisane wyrażeniem
```

	zachłanne	niezachłanne
przykład wyrażenia:	<(.*)>	<(.*?)>
wynik dla <ala>ma<kota>:	ala>ma<kota	ala

```
//Przykład: przeszukuje html w poszukiwaniu adresów
string html = /* wczytaj stronę html */;
regex mail("\\href=\"mailto:(.*?)\">", regex_constants::icase);
smatch what;//Przechowuje wyniki wyszukiwania
if( regex_search(html,what,mail) )
  //what[0] zawiera napis pasujący do wyrażenia
  cout << "wczytany email to" << what[1] << endl;
```

Powtórzenie

- stałość
- współbieżność
  - wątki
  - blokady (mutex), nieblokujące wejście do sekcji krytycznej, zakleszczenia
  - std::atomic, algorytmy lock-free, modele synchronizacji pamięci
  - aktywny obiekt, std::async
  - stl, algorytmy równoległe i wektorowe
- szablony, biblioteka STL
- biblioteka Boost Graph Library
- przetwarzanie tekstu w C++.

Zadanie: problem z kontenerem

Przedstawiony kod jest błędny. Dla N = 5 (linia 4) działa on poprawnie, natomiast po zmianie na N = 500 występują błędy w linii 10 lub 11. Co Twoim zdaniem jest źródłem problemu?

```
vector<Foo> kolekcja;
typedef vector<Foo>::const_iterator Id;
map<int,Id> index;
static const int N = 500;
for(int i=0; i < N; ++i ) {
  Id id = kolekcja.insert( kolekcja.end(), Foo(i) );
  index.insert( make_pair(i,id) );
}
Id id_0 = (*index.find(0)).second;
const Foo& f = *id_0;
cout << f.i_ << endl;
```

Wyścig?

```
using Counter = int;
struct MTCounter { //wsk. na Counter z mutexem
  MTCounter() : counter(new Counter(0)) {}
  void inc() { lock_guard lock(m_); *counter += 1; }
  int get() { lock_guard lock(m_); return *counter; }
  shared_ptr<Counter> counter; //wskaźnik
  mutex m;
};
struct Thread {
  Thread(MTCounter counter) : c(counter) {}
  void operator()() { for(int i=0;i<1000000;++i) c.inc(); }
  MTCounter c;
};
int main() {
  MTCounter counter; Thread t1(counter), t2(counter);
  thread thrd1(ref(t1)), thrd2(ref(t2));
  thrd1.join(); thrd2.join();
  return 0;
}
```

Rozwiązanie

```
//Rozwiązanie - sekcja krytyczna
struct CounterSync {
  CounterSync() : value(0) {}
  int value; mutex m;
};
struct MTCounterSync {
  MTCounterSync() : counter(new CounterSync) {}
  void inc() { lock_buard lock(counter->m); counter->value += 1; }
  int get(){ lock_guard(counter->m); return counter->value; }
  shared_ptr<CounterSync> counter; //współdzielony licznik
};
//Rozwiązanie - licznik wykorzystuje operacje atomowe
using CounterAtomic = std::atomic<int>;
struct MTCounterAtomic {
  MTCounterSync() : counter(new CounterAtomic(0)) {}
  void inc() { *counter += 1; }
  int get(){ return *counter; }
  shared_ptr<CounterAtomic> counter; //współdzielony licznik
};
```

znajomość kodu: podaj wydruk

```
void fxx(int& count, const string& s, int offset) {
  count += s.size() + offset;
}

using PF = std::function<void (int)>;
int x = 0;
vector<PF> vpf;
vpf.push_back([&](int i){fxx(x, string("A"), i);});
vpf.push_back([&](int i){fxx(x, string("AA"), i);});
vpf.push_back([&](int i){fxx(x, string("AAA"), i);});
for_each( vpf.begin(), vpf.end(), [](PF& pf){pf(1);});
cout << x << endl;
```

Podsumowanie

KISS (Keep it simple software)

BUZI (bez udziwnień zbędnych idioto)

Dziękuję