laboratorium: zadanie 12 – dodatkowe termin: 12–16 czerwca 2023 r.

KURS JĘZYKA C++

LAMBDY

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Prolog.

Lambda w programowaniu jest definicją funkcji anonimowej, która nie posiada swojej nazwy. Często lambdy są argumentami innych funkcji. Jeśli funkcja będzie użyta tylko w jednym miejscu, to użycie funkcji anonimowej może być syntaktycznie wygodniejsze niż definiowanie funkcji nazwanej.

Funkcje anonimowe są wszechobecne w funkcyjnych językach programowania. W języku C++ lambdy to po prostu anonimowe funktory, czyli obiekty funkcyjne. Zostały po raz pierwszy wprowadzone w C++11, aby umożliwić definiowane funkcji bezpośrednio w miejscu jej użycia; zwykle lambda jest używana jako parametr innej funkcji, oczekującej wskaźnika do funkcji lub funktora (obiektu wywoływalnego).

Zadanie 1.

Zdefiniuj funktor (klasę albo strukturę z operatorem funkcyjnym), który będzie sumował podawane mu elementy. Podstawową funkcjonalnością tego funktora powinno być podanie bieżącej sumy oraz dodatkowo liczby zsumowanych elementów i średniej arytmetycznej.

Przetestuj działanie funktora podając mu wartości z kilku różnych źródeł danych po kolei: `set<int>`, `list<float>`, `vector<double>`. Zastosuj algorytm `for_each`.

Zadanie 2.

Zdefiniuj szablon obiektu funkcyjnego do obliczania wartości funkcji liniowej $ax+b$. Wykorzystaj w definicji obiekty funkcyjne `multiplies<>` i `plus<>` oraz włącz je do obliczeń korzystając z szablonu wywołującego funkcje `bind<>`.

Przetestuj działanie obiektu funkcyjnego wygenerowanego na podstawie szablonu dla danych typu `int`, `float` i `double`. Czy Twoja definicja zadziała z liczbami typu `complex`?

Zadanie 3.

Zdefiniuj szablon dla bezargumentowej lambdy generującej losowe liczby całkowite typu `int` z określonego zakresu (zakres określ za pomocą parametrów szablonu).

Następnie przetestuj działane generatora wpisując losowe liczby do zwykłaj tablicy `int[N]`, do opakowania na tablicę `array<int, N>` oraz do wektora `vector<int>(N)`. Wpisz losowe liczby korzystając z funkcji `generate` albo `generate_n` i posługując się zdefiniowanym wcześniej generatorem. Po wygenerowaniu danych wypisz zawartość każdej kolekcji funkcją `for_each` (wypisywanie zrealizuj za pomocą jednoargumentowej lambdy `[](int x){...}`).

Na koniec z każdej kolekcji liczby, które są pierwsze i umieść je we wcześniej przygotowanej tablicy za pomocą funcji `copy_if`. Wykorzystaj w tym celu lambdę predykatową, sprawdzającą czy podana liczba całkowita jest pierwsza; lambda ta powinna tak działać aby pozostawić na zewnątrz informację o liczbie znalezionych i przekopiowanych liczb pierwszych. Oczywiście wybrane liczby pierwsze też należy wypisać.

Zadanie 4.

Zdefiniuj kopiowalną klasę `trojkat` reprezentującą trójkąt: trójkąt ma być opisany długościami trzech boków typu `double`, trójkąt ma posiadać nazwę typu `string`, boki trójkąta to pola prywatne, udostępnij jedynie gettery do odczytu długości boków. Boki trójkąta mają być ustawiane w liście inicjalizacyjnej konstruktora a w treści konstruktora należy sprwadzić, czy spełniony jest warunek trójkąta (jeśli nie, to trzeba zgłosić wyjątek); konstruktor bezargumentowy ustawia wszystkie boki na długość 1. W trójkącie zdefiniuj jeszcze dwie funkcje składowe: jedna ma wyliczać obwód trójkąta a druga jego pole (wzór Herona). Na pewno przyda się operator strumieniowy do wypisywania stanu trójkąta.

W wektorze `vector<trojkat>` umieść arbitralnie kilkanaście różnych trójkątów. Następnie wykonaj następujące czynności posługując się algorytmami z biblioteki standardowej i lambdami:

- posortuj trójkąty węglug długości obwodu;
- wskaż trójkąt o maksymalnym i minimalnym polu;
- wypisz tylko trójkąty ostrokątne.

Zadanie 5.

Zdefiniuj rekurencyjną lambdę, która wyznaczy ciąg Collatza zaczynający się od wartości naturalnej $n > 0$ (dla $n = 1$ długość ciągu wynosi 1). W trakcie obliczania długości ciągu, jako efekt uboczny wypisuj kolejne wartości elementów tego ciągu.

Elementy w programie, na które należy zwracać uwagę.

- Definicja funktora jako własnej klasy.
- Definicja szablonu klasy funkcyjnej.
- Definicje lambd bezargumentowych i z argumentami.
- Zastosowanie lambd w algorytmach z STL.
- Definicja lambdy rekurencyjnej.