laboratorium: zadanie 9 termin: 15–19 maja 2023 r.

KURS JĘZYKA C++

LISTA JEDNOKIERUNKOWA

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Prolog.

Szablony w C++ umożliwiają programowanie uogólnione, czyli tworzenie abstrakcyjnych algorytmów oraz struktur danych niezależnych od konkretnych typów, na których one pracują. Typ ten jest określany (w sposób jawny bądź niejawny) w miejscu użycia szablonu — wówczas kompilator na podstawie szablonu wygeneruje odpowiednią definicję funkcji czy klasy z określonym już typem.

Zadanie.

Zdefiniuj szablon klasy `list<T>` dla listy jednokierunkowej w przestrzeni nazw `calc`. Klasa reprezentująca listę ma być napisana zgodnie ze sztuką programowania dynamicznych struktur danych — w pełni funkcjonalny węzeł listy `node<T>` zdefiniuj jako prywatną klasę zagnieżdżoną w opakowaniu `list<T>`, posiadającym wygodny dla programisty interfejs z następującymi operacjami:

(i) wstawienie elementu na zadaną pozycję;

(ii) wstawienie elementu na początek listy (czyli na pozycję 0);

(iii) wstawienie elementu na koniec listy;

(iv) usunięcie elementu z określonej pozycji;

(v) usunięcie elementu z początku listy (czyli z pozycji 0);

(vi) usunięcie elementu z końca listy;

(vii) usunięcie elementu o zadanej wartości (pierwszego od początku);

(viii) usunięcie wszystkich elementów o zadanej wartości;

(ix) określenie pozycji elementu o zadanej wartości (pierwszego od początku);

(x) policzenie wszystkich elelemtów o zadanej wartości;

(xi) zliczenie wszystkich elementów;

(xii) sprawdzenie czy lista jest pusta.

Obiekty list mają być kopiowalne (konstruktor i przypisanie kopiujące i przenoszące). Uzupełnij definicję szablonu o inicjalizację za pomocą listy wartości `initializer_list<T>`. Nie zapomnij o operatorze strumieniowym do wypisania zawartości listy.

Następnie zdefiniuj szablony klas implementujących kolejkę i stos na liście. Klasy te powinny dziedziczyć niepublicznie po liście. Próba pobrania elementu z pustej kolejki albo z pustego stosu ma skutkować zgłoszeniem wyjątku. Definicje te także umieść w przestrzeni nazw `calc`.

Dalej, w przestrzeni nazw `calc`, zdefiniuj szablony dwóch funkcji do pracy z listami:

(i) funkcja `issorted()` ma sprawdzać, czy lista jest uporządkowana;

(ii) funkcja `sort()` ma sortować elementy na liście.

Szablony tych funkcji powinny posiadać dwa parametry: typ danych przechowywanych w liście oraz trejta implementującego operację porównywania elementów wybranego typu. Trejt ma być parametrem domyślnym w szablonie ustawionym na obiekt zawierający operację tradycyjnego porównywania za pomocą operatora $\leq$. Zdefiniuj też innego trejta implementującego porówywanie za pomocą $\geq$. W trejtach zadbaj o specjalizację dla wskaźników a w szczególności dla wskaźnika typu `const char*`.

Na koniec w fukcji `main()` napisz zestaw testów rzetelnie sprawdzających działanie wszystkich zdefiniowanych klas i funkcji, pracujących na różnych typach danych. Obiekty list, stosów i kolejek, które będą poddawane testowaniu stwórz na stercie operatorem `new`; nie zapomnij zlikwidować ich operatorem `delete` przed zakończeniem programu!

Istotne elementy programu.

- Podział programu na pliki nagłówkowe i pliki źródłowe (osobny plik z funkcją `main()`).
- Użycie przestrzeni nazw `calc`.
- Definicja szablonu klasy dla listy jednokierunkowej wraz z zagnieżdżoną definicja węzła.
- Szablony klas dla stosu i kolejki.
- Szablony funkcji do sortowania danych na liście i do weryfikacji posortowania.
- Definicja trejta realizującego porównania.
- Realizacja specjalizacji dla wskaźników a w szczególności dla wskaźnika typu `const char*` w trejtach.
- Implementacja kopiowania i przenoszenia dla listy.
- Inicjalizacja stanu początkowego listy za pomocą `initializer_list<T>`.
- Destrukcja listy.
- Testy w funkcji `main()`.