Zadanie nr 3: Sprawdzanie testu z arytmetyki

1 Cel zadania

Zadanie wymusza praktyczne przećwiczenia dostosowania formatu i formy wyświetlania informacji dla własnych typów danych. Ma ono pokazać potencjalne możliwości adaptowania operacji na standardowym strumieniu wejściowym i wyjściowym do własnych potrzeb. Ma ono również dać wyobrażenie o mechanizmie hermetyzacji oraz tworzenie właściwego interfejsu do struktur danych.

2 Program zajęć

Materiały pomocnicze do zajęć znajdują się w kartotece `~bk/edu/kpo/zad/z3`. Występujące w dalszej części opisu nazwy podkatalogów odnoszą się do zasobów występujących we wcześniej wspomnianej kartotece.

- *Ocena realizacji zadania z poprzedniego laboratorium* – ocenie podlega poprawność realizacji zadania, styl pisania programu oraz opisy.
- *Ocena przygotowania do bieżących zajęć* – ocenie podlega konstrukcja ogólnego schematu blokowego (lub diagramu czynności) dla aktualnego zadania. Diagram ten nie musi zawierać szczegółowo opisanych wszystkich operacji (patrz wyjaśnienie w rozdziale 6).
- *Modyfikacja programu wg wskazań osoby prowadzącej* – ocenie będzie podlegała poprawność realizacji modyfikacji. Pracę nad modyfikacją programu (wszystkie operacje należy wykonywać na kopii) należy rozpocząć już w trakcie pierwszej fazy laboratorium, gdyż prowadzący nie będzie w stanie ocenić wcześniejszego programu wszystkim jednocześnie.
- *Realizacja wstępnej fazy prac nad nowym zadaniem* – wykorzystując przekazywane parametrów przez referencję należy napisać, opisać i przetestować funkcję, która wczytuje symbol z wejścia standardowego, jak też funkcję, która wypisuje znak na wyjściu standardowym. Prototypy tych funkcji powinny mieć postać:

  ```
  bool WczytajSymbol(Symbol &Sym);
  void WyswietlSymbol(Sybmol Sym);
  ```

  Zakładamy, że pierwsza z funkcji będzie zwracać `true`, gdy wczytanie zakończyło się poprawnie i `false` w przypadku przeciwnym. Pracę nad funkcjami należy rozpocząć już w trakcie pierwszej fazy laboratorium, o ile uda się zakończyć wcześniej pracę nad modyfikacją pierwszego programu, a prowadzący nie zdąży ocenić wcześniejszego programu wszystkim osobom.
- *Ocena realizacji wstępnej wersji programu*

3 Opis zadania

Należy napisać program, który umożliwia sprawdzenie listy działań na symbolach. Lista ta czytana jest z wejścia standardowego. Zakładamy, że w osobnych liniach wpisywane są działania wg schematu:

$$argument_1 \quad operator \quad argument_2 = wynik$$

Zakładamy ponadto, że wykorzystywane są tylko dwuargumentowe operatory arytmetyczne tj. $+$, $-$, $*$, $/$. W trakcie wczytywania kolejnych działań program rozpoznaje je, wykonuje i sprawdza, czy faktycznie wpisany wynik jest zgodny z definicją działania. Następnie wyświetla to co wczytał z dodatkową informacją, czy wynik jest poprawny. Jeśli tak nie jest, to dodatkowo informuje jaki jest właściwy wynik tej operacji. Ponadto program musi zliczać ilość operacji związanych z poszczególnymi operatorami oraz ilość poprawnych i błędnych wyników. Program musi też umieć pomijać te wyrażenia, które są błędnie zapisane i zliczać ile takich wyrażeń wystąpiło. Uznajemy, że wprowadzanie danych zostało zakończone, jeśli użytkownik wcisnął klawisze `Ctrl-D`. Kod generowany przez wciśnięcie kombinacji tych klawiszy jest kodem oznaczającym koniec pliku. W programie należy więc sprawdzić, czy wystąpił koniec *pliku* dla wejścia standardowego (lepszym określeniem jest koniec strumienia).

Po zakończeniu fazy wprowadzania danych należy na wyjściu standardowym wypisać pełną statystykę testu. Powinna ona zawierać następujące informacje:

- ilość wszystkich wyrażeń,
- ilość wyrażeń, które były poprawnie zapisane pod względem składni,
- ilość wyrażeń z poprawnie zapisanym wynikiem,
- procentowy wskaźnik poprawnie zapisanych wyrażeń z poprawnym wynikiem w stosunku do wszystkich wyrażeń, które były poprawnie zapisane pod względem składni,
- ilość operacji dodawania i odejmowania,
- ilość operacji mnożenia i dzielenia.

Mimo przedstawionego opisu celem programu nie jest bezpośrednia interakcja z użytkownikiem, a jedynie sprawdzenie testu, który może być w różny sposób podany na wejście standardowe programu. Jednym z nich jest bezpośrednie wprowadzenie z klawiatury, tak jak to zostało opisane. Opis ten został tak sformułowany dla lepszego zrozumienia istoty działania programu. Innym sposobem jest zapisanie listy działań do pliku i jego przekierowanie na wejście standardowe programu, np.

```
./sprawdzenie_testu  < lista_dzialan.txt
```

lub

```
cat lista_dzialan.txt | ./sprawdzenie_testu
```

Uwaga: Pokazane sposoby wywołania programu nie wymagają żadnych dodatkowych zmian w jego kodzie. W szczególności nie ma potrzeby odwoływania się do operacji na plikach w kodzie programu. Wszystkie operacje muszą być realizowane na wejściu standardowym reprezentowanym przez obiekt `std::cin`.

4 Przykład działania programu

Poniżej przedstawiony przykład wyznacza formę komunikatów i ich forma jest obligatoryjny dla programu tworzonego w ramach niniejszego zadania.

```
diablo> ./sprawdzenie_testu
Start sprawdzianu testu arytmetyki symboli

a + c = b
 Odczytano wyrazenie: a + c = b    Wynik poprawny
a * x = d
 Blad skladni wyrazenia. Niepoprawny drugi argument.
a + d > a
 Blad skladni wyrazenia. Brak znaku '='.
a & a = b
 Blad skladni wyrazenia. Niedozwolony znak operatora.
a *    a =    b
 Odczytano wyrazenie: a * a = b    Wynik niepoprawny. Wlasciwy wynik to: a
b * b =       a
 Odczytano wyrazenie: b * b = a    Wynik poprawny
^D

Statystyka:
              Ilosc wszystkich wyrazen: 6
    Ilosc poprawnie zapisanych wyrazen: 3
    Ilosc wyrazen z poprawnym wynikiem: 2
   Procentowo ilosc poprawnych wynikow: 66.7%
Ilosc operacji odejmowania i dodawania: 1
   Ilosc operacji mnozenia i dzielenia: 2
```

Przykład wywołania *wsadowego* (przekierowanie zawartości pliku na wejście standardowe programu).

```
diablo> cat lista_dzialan.txt
a + c = d
a * x = d
a + d > a
a & a = b
a *    a =    b
b * b =       d
diablo> ./sprawdzenie_testu < lista_dzialan.txt
Start sprawdzianu testu arytmetyki symboli

 Odczytano wyrazenie: a + c = d    Wynik poprawny
 Blad skladni wyrazenia. Niepoprawny drugi argument.
 Blad skladni wyrazenia. Brak znaku '='.
 Blad skladni wyrazenia. Niedozwolony znak operatora.
 Odczytano wyrazenie: a * a = b    Wynik niepoprawny. Wlasciwy wynik to: a
 Odczytano wyrazenie: b * b = d    Wynik poprawny
```

```
Statystyka:
              Ilosc wszystkich wyrazen: 6
    Ilosc poprawnie zapisanych wyrazen: 3
    Ilosc wyrazen z poprawnym wynikiem: 2
   Procentowo ilosc poprawnych wynikow: 66.7%
Ilosc operacji odejmowania i dodawania: 1
   Ilosc operacji mnozenia i dzielenia: 2
```

5 Wymagania co do konstrukcji

Należy pamiętać, że komunikaty o błędach należy kierować na wyjście *standard error* (reprezentowane jest ono przez obiekt `std::cerr`). Na wyjście standardowe reprezentowane przez obiekt `std::cout` kierujemy tylko komunikaty odpowiadające poprawnemu działaniu.

Należy zdefiniować struktury danych modelujące odpowiednio wrażenie algebraiczne oraz zestawienie statystyk. Konieczne jest również zdefiniowanie typu wyliczeniowego reprezentującego operatory (**uwaga**: elementami tego typu nie mogą być znaki `+`, `-`, `*`, `/`; muszą być nimi odpowiednie napisy, np. `op_dodawania`, `op_odejmowania` itd.).

Należy zdefiniować rodzaj *interfejsu* dla zmiennych typów odpowiadających poszczególnym strukturom. Ma on postać funkcji, które wykonują odpowiednie operacje na tych strukturach. Wszystkie funkcje związane z daną strukturą powinny być zdefiniowane w osobnym module, np.

```
WyrazenieAlgeb.hh   <-  definicja struktury WyrazenieAlgeb oraz prototypy
                        funkcji zdefiowane w module.
WyrazenieAlgeb.cpp  <-  definicja funkcji operujących na strukturze WyrazenieAlgeb
```

Przykład prototypów funkcji realizujących odpowiednie operacje dla struktury `WyrazenieAlgeb`.

```
bool InterpretujOperacjeIPorownajWynik( WyrazenieAlgeb  Wyrazenie );
void WstawElementyWyrazenia( WyrazenieAlgeb   &Wyrazenie,
                                    Symbol    Arg1,
                                    Operator  Op,
                                    Symbol    Arg2,
                                    Symbol    Wynik
                           );
```

Przykład prototypów funkcji realizujących odpowiednie operacje dla struktury `Statystyka`.

```
void InkrementujLiczbeOperDodOdej( Statystyka &Stat );
void WyswietlStatystyke( Statystyka  Stat );
```

Uwaga: Poza modułami w pozostałej części programu nie powinno być operacji, które są bezpośrednio wykonywane na polach zmiennej typu `WyrazenieAlgeb` lub `Statystyka`.
Nazwy typów nie są obligatoryjne. Można je dobierać wg własnego uznania. Istotne jest, aby modelowały właściwe pojęcia.
Wszystkie funkcje oraz przeciążenia operatorów muszą być opisane. Wspomniany opis musi zawierać:

1. ogólną informację co dana funkcja robi,

2. warunki wstępne (o ile są konieczne),
3. opis parametrów wywołania funkcji,
4. opis tego co dana funkcja zwraca.

W trakcie kompilacji finalnej wersji programu nie mogą generować się żadne ostrzeżenia. Kompilację należy wykonywać z opcjami `-Wall`**,** `-std=c++11` **oraz** `-pedantic`.

6 Przygotowanie do zajęć

6.1 Tydzień 0

Przed zajęciami należy napisać schemat blokowy (lub diagram czynności) przedstawiający ideę działania programu bez wchodzenia nadmiernie w detale, np. operacja wyświetlenia statystyki powinna być pojedynczym bloczkiem. Ważniejsze jest, aby zastanowić się co się ma stać, gdy nie powiedzie się wczytanie wyrażenia. Należy je rozpatrywać jako całość bez wchodzenia w szczegóły, czy nie powiodło się czytanie pierwszego argumentu, czy też jakiegoś innego elementu tego wyrażenia.

6.2 Tydzień 1

Przed zajęciami muszą zostać przygotowane następujące elementy zadania. Wszystko co będzie ponad to będzie oceniane *in plus*.

- Należy przygotować możliwie szczegółowy schemat blokowy (lub diagram czynności) całego programu. Głównie chodzi o właściwą reakcję na błędy napotykane przy czytaniu wyrażenia. Wyświetlenie statystyki należy potraktować jako pojedynczy bloczek. Podobnie operację czytania symbolu należy również potraktować jako jeden bloczek. Nie wchodzimy w szczegóły jak konkretnie jest to realizowane. Ważne są reakcje na to, czy ta operacja powiodła się, czy też nie.
- Należy napisać osobny schemat blokowy (lub diagram czynności) czytania symbolu z wejścia standardowego, który będzie podstawą realizacji przeciążenia operatora >> dla typu `Symbol`.
- Na podstawie wcześniejszego schematu blokowego (lub diagramu czynności) należy napisać kod przeciążenia operatora czytania symbolu ze strumienia `istream` oraz zapisu do strumienia `ostream`.
- Należy napisać schemat blokowy (lub diagram czynności) dla operacji czytania wyrażenia algebraicznego, który będzie podstawą realizacji przeciążenia operatora >> dla typu `WyrazenieAlgeb`.

6.3 Tydzień 2

Rozliczenie się z gotowego programu i rozpoczęcie następnego zadania (tydzień 0 dla zadania nr 4).