Laboratorium: zadanie nr 10 *22 maja 2006*

JĘZYK PROGRAMOWANIA C++

(DE)SZYFROWANIE DANYCH W STRUMIENIACH

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Zaimplementuj manipulator `encrypt(k)`, który zapewnia, że wyjście na strumień `ostream` będzie kodowane za pomocą klucza `k`. Podobnie zaimplementuj manipulator `decrypt(k)` dla strumienia `istream`, który będzie odkodowywał tekst zakodowany wcześniej kluczem `k`. Zdefiniuj również manipulatory bezparametrowe `noencrypt` dla strumienia wyjściowego i `nodecrypt` dla strumienia wejściowego, wyłączające dalsze kodowanie i dekodowanie.

Proces szyfrowania ma dotyczyć strumieni znakowych. Kodowane lub dekodowane powinny być tylko znaki o kodach z zakresu $32 \ldots 126$. Parametr k powinien być liczbą naturalną nie większą niż 95. Jako funkcję kodującą znak o kodzie z przyjmij $e(z)$:

$$e(z) = \begin{cases} ((z - 32 + \mathtt{k}) \bmod 95) + 32 & \text{dla} \quad z \in \{32 \ldots 126\} \\ z & \text{dla} \quad z \notin \{32 \ldots 126\} \end{cases}$$

Funkcja dekodująca $d(z)$ ma przywracać pierwotną postać znku.

Następnie napisz program, który czyta dane linia po linii ze wskazanego strumienia wejściowego, ewentualnie koduje tekst i posyła go wskazanego strumienia wyjściowego. Sygnałem do rozpoczęcia/zakończenia kodowania jest linia zawierająca pojedynczy znak kropki. Nazwy plików oraz klucz k przekaż do programu poprzez parametry wywołania. Sprawdź poprawność szyfru analogicznym programem dekodującym.

Wskazówka! Operacje zapisu i odczytu nie są wykonywane bezpośrednio przez strumienie, lecz przez specjalne obiekty obsługujące bufory strumieni. Bufor strumienia jest klasą typu `basic_streambuf<>`. Dla typów znakowych jest zdefiniowana specjalizacja o nazwie `streambuf`. Użytkownik może więc zdefiniować swoje bufory dla strumieni typu `istream` (nadpisując metodę `overflow()`) lub `ostream` (nadpisując metody `underflow()` i `uflow()`), które dziedziczą publicznie po `streambuf` i podmienić je metodą `rdbuf()`.

Zdefiniuj własne klasy buforów (osobno dla strumienia wejściowego `istream` i dla strumienia wyjściowego `ostream`) dziedziczące po `streambuf`. Manipulator, który będzie podmieniał te bufory, powinien zapamiętać również wskaźnik na poprzedni bufor w tym strumieniu (jako pole w nowym buforze). Do sprawdzenia, czy mamy do czynienia z oryginalnym buforem czy z naszą podróbką wykorzystaj operator rzutowania `dynamic_cast<>`. Jeśli wyrażenie `dynamic_cast<buf_w*>(rdbuf())` zwróci wskaźnik 0, to znaczy że mamy do czynienia z orygiałem, w przeciwnym przypadku z naszą podróbką (gdzie `buf_w` jest zdefiniowaną przez nas klasą bufora).