Zadania do przedmiotu Programowanie C++
2016/2017

5 kwietnia 2017

1 Standardowe wejście/wyjście, operacje arytmetyczne na zmiennych, instrukcja warunkowa

Zadanie 1. (prosta suma)

Napisz program, który wczytuje ze standardowego wejścia dwie liczby i wypisuje ich sumę.

Zadanie 2. (BMI)

Napisz program, który wczytuje Twoją masę (podaną w kg) i wzrost (podany w cm) i na tej podstawie wylicza i wypisuje BMI (czyli m/h^2 w kg/m^2). Jeśli wyliczona wartość jest mniejsza niż 18.5, program powinien wypisać komunikat: `jestes za chudy`. Jeśli wyliczona wartość jest większa niż 25, program powinien wypisać komunikat `jestes za gruby`.

Zadanie 3. (równanie kwadratowe)

Napisz program rozwiązujący równanie kwadratowe o następującym działaniu:

1. Program wypisuje komunikat `Podaj współczynnik a:`
2. Użytkownik wpisuje wartość współczynnika a.
3. Program wypisuje komunikat `Podaj współczynnik b:`
4. Użytkownik wpisuje wartość współczynnika b.
5. Program wypisuje komunikat `Podaj współczynnik c:`
6. Użytkownik wpisuje wartość współczynnika c.
7. Program wypisuje rozwiązania równania $ax^2 + bx + c = 0$.

Twój program powinien radzić sobie również z przypadkami zdegenerowanymi, np. gdy podane współczynniki odpowiadają równaniu liniowemu ($a = 0$).

Zadanie 4. (Masa relatywistyczna)

Napisz program wyznaczający relatywistyczny wzrost masy obiektu poruszającego się z prędkością v (niekoniecznie $v \ll c$), pobieranej z klawiatury. Przetestuj program dla kilku reprezentatywnych wartości testowych.

Zadanie 5. (Starszeństwo)

Poproś użytkownika o podanie wieku dwóch osób i wskaż, która z nich jest starsza. Jeśli obie osoby mają powyżej 100 lat, program powinien zachować się w szczególny sposób.

Zadanie 6. (Prosty kalkulator)

Napisz niewielki kalkulator, który pobiera na wejściu jeden z operatorów arytmetycznych oraz dwa argumenty, po czym wyświetla wynik obliczeń otrzymany na podstawie tych danych.

Zadanie 7. (Prosty kalkulator 2)

Składnia języka C++ udostępnia przydatną w niektórych sytuacjach instrukcję wyboru *switch-case*. Zapoznaj się ze składnią instrukcji wyboru przy wykorzystaniu książek lub Internetu. Następnie napisz niewielki kalkulator, który pobiera na wejściu jeden z operatorów arytmetycznych oraz dwa argumenty, po czym wyświetla wynik obliczeń otrzymany na podstawie tych danych.

Zadanie 8. (Rok przestępny)

Napisz program wczytujący z klawiatury liczbę całkowitą reprezentującą rok, a następnie wypisujący informację o tym, czy jest to rok przestępny, czy nie. *Wskazówka:* Operator reszty z dzielenia to %.

Zadanie 9. (Wymiary walca)

Napisz program, który pobierze od użytkownika dwie dodatnie liczby rzeczywiste charakteryzujące wymiary walca - promień podstawy oraz wysokość, a następnie wypisze na ekran wartość pola powierzchni walca i jego objętość. Wartość liczby π występuje jako stała M_PI w bibliotece ¡cmath¿. Należy sprawdzić, czy podane przez użytkownika liczby są większe od zera, a w przeciwnym wypadku wypisać odpowiedni komunikat.

Zadanie 10. (Weryfikacja hasła)

Zaimplementuj prosty system weryfikacji haseł. Po uruchomieniu program ten przyzna dostęp wyłącznie użytkownikowi o nazwie *admin*, który dysponuje właściwym hasłem.

Zadanie 11. (Weryfikacja hasła 2)

Rozszerz program kontrolujący hasła w taki sposób, aby akceptował wielu użytkowników, z których każdy ma swoje hasło. Zagwarantuj, aby właściwe hasła były przypisane właściwym użytkownikom. Udostępnij możliwość ponownego zalogowania użytkownika, jeśli pierwsza próba nie powiodła się. Zastanów się, jak łatwo (lub trudno) można zrealizować taką funkcjonalność w przypadku dużej liczby użytkowników i haseł.

Zadanie 12. (Weryfikacja hasła 3)

Pomyśl o tym, jakie elementy lub funkcje języka ułatwiłyby dodawanie nowych użytkowników bez potrzeby ponownej kompilacji programu weryfikującego hasła

(nie czuj się zmuszony do rozwiązania tego problemu za pomocą funkcji C++, które poznałeś na zajęciach. Możesz powrócić do tego zadania, gdy poznasz odpowiednie elementy języka na kolejnych zajęciach.

Zadanie 13. (Trójka pitagorejska)

(Tomasz Tarkowski) Napisz program, który wczyta trzy liczby typu `int` a następnie sprawdzi czy stanowią one trójkę pitagorejską, to znaczy czy suma kwadratów dwóch mniejszych liczb jest równa kwadratowi liczby największej. Liczby 3, 4 oraz 5 stanowią taką trójkę. W przypadku podania trójki pitagorejskiej należy wyświetlić komunikat: `Liczby stanowia trojke pitagorejska.`, w przeciwnym razie powinien być to komunikat: `To nie jest trojka pitagorejska.` Uwaga: W programie należy założyć, że użytkownik wpisze liczby w dowolnej kolejności, np. 5, 3, 4.

2 Pętle

Zadanie 14. (Silnia)

(Tomasz Tarkowski) Napisz program obliczający silnię. Niech program prosi użytkownika o podanie liczby całkowitej. Następnie program powinien sprawdzić czy podana liczba jest nieujemna. Jeśli jest ujemna to program powinien wyświetlić komunikat: `Blad. Liczba jest ujemna.` i zakończyć działanie. Jeśli liczba jest równa zero to program powinien wypisać wartość silni z zera czyli liczbę 1. Jeśli wartość jest większa od zera to program powinien obliczyć wartość silni z tej liczby z użyciem pętli „for" i następnie wypisać obliczoną wartość. Schemat programu jest następujący (należy zamienić komentarze `/* ... */` rzeczywistym kodem):

```
/* ... */

if (n < 0)
{
  /* ... */
}
else if (n == 0)
{
  /* ... */
}
else
{
  /* ... */
  for (/* ... */)
  {
    /* ... */
  }
}

/* ... */
```

Zadanie 15. (Liczby parzyste z przedziału $[a, b]$)

(Tomasz Tarkowski) Uzupełnij poniższy kod tak, aby wyświetlił liczby parzyste z przedziału $[a, b]$ podanego przez użytkownika programu.

```
/* ... */

int a, b;
cin >> a >> b;

for (int i = a; i <= b; ++i)
{
```

```
  if (/* ... */)
  {
    cout << i << " ";
  }
}
cout << endl;

/* ... */
```

Zadanie 16. (Sumowanie szeregu – funkcja cos)

(Tomasz Tarkowski) Funkcję cos można przedstawić w postaci szeregu:

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots$$

Napisz program obliczający wartość funkcji cos dla argumentu x wczytywanego z klawiatury. Obliczanie powinno odbywać się poprzez sumowanie szeregu. Sumowanie powinno zakończyć się gdy kolejny wyraz sumy ma moduł mniejszy niż wybrana precyzja obliczeń. Przyda się funkcja `abs()` z pliku `cmath`, która oblicza moduł (wartość bezwzględną) liczby:

```
#include <cmath>
/* ... */
double modul_z_y = abs(y)
```

Dla porównania oblicz wartość $\cos(x)$ z użyciem funkcji `cos()` z pliku `cmath`:

```
#include <cmath>
/* ... */
cout << "Powinno wyjsc (okolo): " << cos(x) << endl;
```

Zadanie 17. (Trójkąty połączone bokami)

Napisz program, który wykorzystuje tylko dwa polecenia wyjściowe *cout*, aby wygenerować wzór złożony z symboli kratki # ułożonych w kształt dwóch trójkątów o boku N połączonych ze sobą bokami.

Zadanie 18. (Łamacz haseł)

Napisz program służący do weryfikacji haseł, który pobiera od użytkownika login i hasło aż do momentu, gdy wpisane dane umożliwią dostęp do systemu.

Zadanie 19. (Liczby pierwsze)

Napisz program sprawdzający, czy dana liczba naturalna jest pierwsza. Program powinien wczytywać liczbę ze standardowego wejścia i drukować na standardowe wyjście odpowiedni komunikat.

Zadanie 20. (Ankieter *)

Napisz program udostępniający opcję sumowania wyników ankiety, w której mogą wystąpić trzy różne wartości. Dane wejściowe wprowadzane do programu to pytanie ankietowe oraz trzy możliwe odpowiedzi. Pierwszej odpowiedzi przypisywana jest wartość 1, drugiej 2, a trzeciej 3. Odpowiedzi są sumowane do chwili, w której użytkownik wprowadzi 0 – wtedy program powinien pokazać wyniki ankiety. Postaraj się wygenerować wykres słupkowy, pokazujący wyniki przeskalowane w taki sposób, aby zmieściły się na ekranie bez względu na liczbę udzielonych odpowiedzi.

Zadanie 21. (Łamacz haseł 2)

Zmodyfikuj program służący do weryfikacji haseł tak, aby dawał użytkownikowi tylko kilka szans na podanie poprawnego hasła (użycie łamacza haseł będzie trudne).

Zadanie 22. (Rozkład na czynniki pierwsze)

Napisz program rozkładający daną liczbę naturalną na czynniki pierwsze w następujący sposób. Sprawdzamy, czy liczba dzieli się przez 2. Jeżeli tak, to stwierdzamy, że dwójka występuje w jej rozkładzie na czynniki pierwsze, a samą liczbę dzielimy przez 2. Czynność tę powtarzamy, aż liczba przestanie być podzielna przez 2. Następnie powtarzamy tę samą procedurę badając podzielność przez 3, 4, itd., aż rozważana liczba stanie się równa 1.

Zadanie 23. (Sumowanie szeregu – inne funkcje)

Napisz program obliczający wartość innych funkcji (np. sin, exp, ln) metodą sumowania odpowiedniego szeregu. Sumowanie powinno zakończyć się po zsumowaniu liczby wyrazów wczytanej przez użytkownika. Zbadaj, ile wyrazów szeregu należy sumować, aby uzyskać dokładność wyniku na poziomie 1%.

Zadanie 24. (Arytmetyka binarna)

Jeśli znasz arytmetykę binarną i wiesz, w jaki sposób można zamieniać liczby dziesiętne na binarne (i odwrotnie), spróbuj napisać programy, które wykonują takie konwersje dla liczb o nieograniczonej długości (możesz założyć, że są na tyle małe, iż mogą być przechowywane w standardowym typie *int* języka C++.

Zadanie 25. (Liczby heksadecymalne *)

Czy znasz liczby heksadecymalne? Napisz program, który pozwoli użytkownikowi na podanie liczby w systemie binarnym, dziesiętnym lub heksadecymalnym, a następnie wyświetli ją w każdym z nich.

Zadanie 26. (Konwerter systemów liczbowych **)

Potrzebujesz dodatkowego wyzwania? Uogólnij kod poprzedniego zadania w taki sposób, aby stworzyć program, który przekształca dowolna liczbę z systemu szesnastkowego lub niższego na liczbę opartą na innym systemie.

Zadanie 27. (Rozkład liczb na czynniki)

Napisz program, który rozkłada podaną przez użytkownika liczbę całkowitą na czynniki pierwsze. Przykładowe działanie programu:

```
[program]    Podaj liczbe calkowita:
[użytkownik] 24
[program]    24 = 2^3 * 3^1
```

3 Liczby losowe, tablice, vector

Zadanie 28. (Rzuty kostką sześcienną)

(Tomasz Tarkowski) Napisz program symulujący serię rzutów kostką sześcienną. Niech program prosi o podanie liczby rzutów a następnie wypisze rezultat losowania. Zadbaj o niedeterministyczne zachowanie programu. Przykładowa sesja mogłaby wyglądać następująco:

```
Podaj liczbe rzutow: 8
Wylosowano: 4 2 2 6 1 3 4 1
```

Zadanie 29. (Znajdowanie minimum w tablicy liczb losowych)

(Tomasz Tarkowski) Napisz program tworzący w pamięci komputera 10-elementową tablicę liczb typu `double` i zapełnij ją (z użyciem pęti „for”) liczbami losowymi z przedziału $[0, 1[$. Następnie (z użyciem drugiej pętli „for”) wypisz zawartość tablicy. Na koniec uzupełnij kod programu tak, aby program znajdował i wypisywał indeks najmniejszego elementu.

Zadanie 30. (Sortowanie bąbelkowe)

(Tomasz Tarkowski) Poniżej znajduje się program, który sortuje tablicę metodą bąbelkową (sortowanie bąbelkowe).

```
#include <iostream>
#include <cstdlib>
using namespace std;

// Sortowanie babelkowe

int main()
{
  int n = 10;
  double tab[n];

  for ( int i = 0; i < n; ++i )
    tab[i] = ( rand() / ( RAND_MAX + 1. ));

  cout << "Przed sortowaniem:" << endl;
  for (int i = 0; i < n; ++i)
    cout << tab[i] << endl;

  bool zmiana;

  do {
    zmiana = false;
```

```
    for (int i = 0; i < n-1; ++i) {
      if (tab[i] > tab[i+1]) {
        zmiana = true;
        double temp = tab[i];
        tab[i] = tab[i+1];
        tab[i+1] = temp;
      }
    }
  } while (zmiana);

  cout << "Po sortowaniu:" << endl;
  for (int i = 0; i < n; ++i)
    cout << tab[i] << endl;

  return 0;
}
```

Zmodyfikuj program tak, aby dla każdej iteracji pętli „do-while" wyświetlał listę par indeksów elementów ulegających zamianie. Oto jak może wyglądać rezultat działania takiego programu:

```
(0,1) (1,2) (2,3) (4,5) (5,6) (6,7) (7,8) (8,9)
(3,4) (4,5) (5,6) (6,7) (7,8)
(2,3) (3,4) (4,5) (5,6) (6,7)
(1,2) (2,3) (3,4) (4,5) (5,6)
(0,1) (1,2) (3,4) (4,5)
(2,3)
(1,2)
```

Każdy wiersz odpowiada jednej iteracji pętli do-while. Na przykład wiersz:

```
(0,1) (1,2) (3,4) (4,5)
```

informuje, że zostały zamienione elementy o indeksach najpierw 0 i 1, następnie 1 i 2, dalej 3 i 4, na koniec 4 i 5.

Zadanie 31. (Sito Eratostenesa)

Napisz program, który szuka liczb pierwszych w przedziale od 1 do n metodą sita Eratostenesa (opis algorytmu dostępny na Wikipedii). Przykładowe działanie programu:

```
[program]    Podaj koniec zakresu szukania liczb pierwszych:
[użytkownik] 10
[program]    W przedziale 1-10 znajduje sie 4 liczby pierwsze:
[program]    2
[program]    3
[program]    5
[program]    7
```

Zadanie 32. (Hipoteza Goldbacha)

Według hipotezy Goldbacha każda parzysta liczba naturalna większa od dwóch jest sumą dwóch liczb pierwszych. Napisz program testujący tę hipotezę. Dla kolejnych testowanych liczb powinien wypisywać odpowiedni rozkład, np.:

```
4=2+2
6=3+3
8=5+3
10=7+3
```

Jeżeli dla jakiejś liczby parzystej odpowiedni rozkład nie został znaleziony, program powinien wypisać `Hipoteza Goldbacha obalona!!!`.

Zadanie 33. (Dwie postaci szeregu)

Szereg 10-elementowy można przedstawić w postaci *tradycyjnej*:

$$a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + a_6x^6 + a_7x^7 + a_8x^8 + a_9x^9$$

albo w postaci *nawiasowej*:

$$a_0 + x(a_1 + x(a_2 + x(a_3 + x(a_4 + x(a_5 + x(a_6 + x(a_7 + x(a_8 + xa_9))))))))).$$

Analogicznie można przedstawić szereg o dowolnej liczbie elementów.

Obliczenia dla szeregu w postaci nawiasowej realizuje się od najbardziej wewnętrznego nawiasu. W przypadku 10-elementowym wygląda to następująco: a) bierzemy wartość a_9, b) mnożymy ją przez x, c) dodajemy a_8, d) mnożymy przez x itd. Na końcu trzeba dodać a_0.

Napisz program obliczający sumę szeregu **30**-elementowego. Zadeklaruj tablicę:

```
double a[30];
```

i zapełnij ją liczbami losowymi z przedziału $[0, 1[$. Załóż, że i-ty element tablicy ma znaczenie współczynnika a_i szeregu 30-elementowego. Program powinien obliczać sumę szeregu korzystając z reprezentacji nawiasowej. Niedozwolone jest korzystanie z funkcji `pow()`. W odpowiedni sposób wykorzystaj pętlę „for”. Wynik obliczeń powinien być wyświetlany na ekranie. Zadbaj o niedeterministyczne zachowanie programu (różne uruchomienia programu generują różną zawartość tablicy).

Zadanie 34. (Orzeł czy reszka?)

Napisz program symulujący rzut monetą. Uruchom go wiele razy (tzn. wiele losowań w pętli). Czy uzyskane wyniki wyglądają Twoim zdaniem na losowe?

Zadanie 35. (Element najmniejszy)

Napisz program znajdujący indeks najmniejszego elementu tablicy. Program powinien wczytywać długość tablicy, tworzyć tę tablicę w pamięci, wypełniać losowymi liczbami rzeczywistymi z przedziału od 0 do 1, a następnie wypisywać elementy tablicy wraz z indeksami oraz znaleziony indeks i wartość elementu najmniejszego.

Zadanie 36. (Obliczenia statystyczne)

Niech dana będzie próba losowa N wartości x_i. Średnia i odchylenie standardowe z tej próby wynoszą odpowiednio:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \sigma_x = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2}$$

Napisz program, który pobiera ze standardowego wejścia dodatnie liczby rzeczywiste aż do momentu, gdy użytkownik wpisze wartość -1, a następnie wyznacza średnią i odchylenie standardowe podanych liczb. *Wskazówka*: użyj pojemnika ¡vector¿ ze standardowej biblioteki szablonów STL.

Zadanie 37. (Dominanta sondażowa)

Dominantą zbioru danych w statystyce nazywamy taką wartość, która występuje w nim najczęściej. Napisz kod, który przetwarza tablicę danych sondażowych, aby ustalić ich dominantę. Odpowiedź na pytanie sondażowe polegała na podawaniu liczby z zakresu od 1 do 10. W sytuacji gdy istnieje wiele dominant, można wybrać dowolną z nich. Program powinien wczytywać odpowiedzi na pytanie ankietowe do momentu, w którym użytkownik wprowadzi wartość 0.

Zadanie 38. (Wicelider)

Napisz program znajdujący położenie drugiego co do wielkości elementu tablicy. Program powinien wypełniać tablicę o zadanej długości losowymi liczbami rzeczywistymi z przedziału od 0 do 1, a następnie wypisywać elementy tablicy wraz z indeksami oraz znaleziony indeks i wartość elementu drugiego co do wartości.

Zadanie 39. (Losowanie Lotto *)

Napisz program losujący 6 parami różnych liczb naturalnych z przedziału od 1 do 49 włącznie i wypisujący je w kolejności rosnącej. W programie nie używaj sortowania. *Wskazówka*: użyj pojemnika ¡set¿ ze standardowej biblioteki szablonów STL.

Zadanie 40. (Szyfr podstawieniowy)

Napisz program używający szyfru podstawieniowego, w którym wszystkie wiadomości składają się z wielkich liter i znaków interpunkcyjnych. Pierwotna wiadomość jest zwana tekstem jawnym, zaś szyfrogram tworzy się poprzez podmianę każdej z liter na inną. Utwórz w programie tablicę typu *const* składającą się z 26 elementów *char* służących do szyfrowania. Program powinien odczytywać tekst jawny i wyprowadzać odpowiadający mu szyfrogram.

Zadanie 41. (Szyfr podstawieniowy 2 *)

Zmodyfikuj powyższy program w taki sposób, by konwertował szyfrogram z powrotem na tekst jawny w celu zweryfikowania kodowania i dekodowania.

Zadanie 42. (Szyfr podstawieniowy 3 **)

Aby jeszcze bardziej utrudnić problem szyfru podstawieniowego, zmodyfikuj program w taki sposób, by zamiast wykorzystywać wbudowaną tablicy wartości *const*, generował w sposób losowy wzorzec szyfrowania. W praktyce oznacza to umieszczenie losowych znaków w każdym elemencie tablicy. Pamiętaj, że dana litera nie może być substytutem samej siebie. Nie możesz także użyć tej samej litery dwukrotnie.

4 Funkcje, rekurencja

Zadanie 43. (Silnia)

(Tomasz Tarkowski) Uzupełnij poniższy kod tak, aby rezultatem działania programu było obliczenie silni z danej liczby. Załóż, że użytkownik programu poda dodatnią liczbę całkowitą.

```
#include <iostream>
using namespace std;

int silnia(int m)
{
  /* ... */
}

int main()
{
  cout << "Podaj dodatnia liczbe calkowita: ";
  int n;
  cin >> n;
  cout << n << "! = " << silnia(n) << endl;
  return 0;
}
```

Zadanie 44. (Podwójna silnia)

(Tomasz Tarkowski) Silnia podwójna $n!!$ jest zdefiniowana jako:

$$\begin{aligned} 1 \quad &\text{dla} \quad n = 0, 1, \\ n \cdot (n-2)!! \quad &\text{dla} \quad n \geqslant 2. \end{aligned}$$

Na przykład:

$$7!! = 7 * 5 * 3 * 1 = 105.$$

Napisz program w którym zdefiniujesz dwie funkcje:

- `silnia_2_r` obliczającą silnię podwójną w sposób rekurencyjny,
- `silnia_2_i` obliczającą silnię podwójną w sposób iteracyjny.

W funkcji `main()` wywołaj obydwie funkcje dla liczby podanej przez użytkownika i porównaj wyniki obliczeń (powinno wyjść tyle samo). Sprawdź czy dla 7 otrzymujesz wartość 105.

Zadanie 45. (Algorytm Euklidesa)

Napisz funkcję znajdującą największy wspólny dzielnik dwóch liczb naturalnych metodą Euklidesa. Mając dwie liczby, większą z nich zastępujemy resztą z dzielenia przez mniejszą. Procedurę powtarzamy, aż jedna z liczb stanie się równa zero. Wtedy druga jest poszukiwanym wspólnym dzielnikiem wyjściowych liczb.

Zadanie 46. (Ciąg Fibonacciego)

Ciąg Fibonacciego to ciąg liczb, w którym pierwszy wyraz jest równy 0, drugi jest równy 1 a każdy następny jest sumą dwóch poprzednich:

$$F_0 = 1 \qquad F_1 = 1 \qquad F_n = F_{n-1} + F_{n-2}$$

Napisz program wyznaczający n-ty wyraz ciągu najpierw przy użyciu funkcji iteracyjnej, a następnie rekurencji. Które podejście jest łatwiejsze?

Zadanie 47. (Czy posortowana?)

Napisz funkcję typu *bool*, której argumentami są tablica oraz liczba jej elementów. Funkcja powinna ustalać, czy dane w tablicy są posortowane. Rozwiązanie powinno wymagać użycia tylko pojedynczej pętli (bez zagnieżdżania).

Zadanie 48. (Szukanie sekwencyjne)

Napisz funkcję, która otrzymuje tablicę liczb całkowitych oraz wartość poszukiwaną i zwraca liczbę wystąpień tej wartości w podanej tablicy. Rozwiąż problem najpierw przy użyciu iteracji, a następnie rekurencji.

Zadanie 49. (Sortowanie przez scalanie *)

Sortowanie przez scalanie to rekurencyjny algorytm sortowania danych, stosujący metodę *dziel i zwyciężaj*, o złożoności czasowej lepszej niż dla poznanych poprzednio algorytmów sortowania. Opis działania algorytmu jest przedstawiony np. na stronie:

`http://main.edu.pl/pl/user.phtml?op=lesson&n=24&page=algorytmika`

Korzystając z zamieszczonego tam opisu, postaraj się zaimplementować program sortujący, korzystając z funkcji dokonującej podziału oraz złączającej posortowane ciągi liczb całkowitych. W razie problemów spróbuj zrozumieć zamieszczone rozwiązanie.

Zadanie 50. (Ocenianie kształtujące)

W szkolnych klasach 1–3 popularną metodą oceniania sprawdzianów jest ich samodzielne sprawdzanie przez dzieci przez porównanie z odpowiedzią wzorcową na tablicy (małe dzieci nie oszukują). Napisz funkcję, która oblicza liczbę

błędów popełnioną przez dziecko podczas pisania dyktanda. Funkcja powinna przyjmować jako argumenty dwa łańcuchy tekstowe – jeden napisany przez nauczyciela, drugi przez ucznia. Zweryfikuj działanie funkcji na przykładowym tekście wprowadzanym z klawiatury.

Zadanie 51. (Rekurencyjna potęga)

Napisz program w sposób rekurencyjny obliczający funkcję potęgową x^y.

Zadanie 52. (Parzystość łańcucha binarnego *)

Rozważ tablicę reprezentującą łańcuch binarny, w której elementy mają wartości 0 lub 1. Napisz funkcję typu *bool* ustalającą, czy łańcuch binarny jest nieparzysty (czyli posiada nieparzystą liczbę bitów równych 1). *Wskazówka*: pamiętaj, że funkcja rekurencyjna będzie zwracać true (nieparzystość) lub false (parzystość), a nie liczbę bitów równych 1. Rozwiąż problem najpierw przy użyciu iteracji, a następnie rekurencji.

Zadanie 53. (Rekurencyjne szukanie (binarne) *)

Napisz funkcję rekurencyjną, która pobiera posortowaną tablicę, element docelowy oraz odszukuje ten element w tablicy (jeśli nie znajdzie elementu, powinna zwrócić -1). Jak szybkie może być takie szukanie? Czy można osiągnąć lepszy wynik bez potrzeby sprawdzania każdego elementu?

5 Wskaźniki, referencje

Zadanie 54. (Przekazywanie tablic do funkcji)

(Tomasz Tarkowski) Napisz program tworzący w pamięci komputera 10-elementową tablicę liczb typu `double` i zapełnij ją zerami w następujący sposób:

```
double t[10] = {0., 0., 0., 0., 0., 0., 0., 0., 0., 0.};
```

Napisz funkcję `losowanie`:

```
void losowanie(double* tab, int n)
```

która zapełnia tablicę `tab` n liczbami losowymi z przedziału $[0, 1[$. Napisz funkcję `drukowanie`:

```
void drukowanie(double* tab, int n)
```

która wypisuje zawartość n-elementowej tablicy `tab` na ekran. Napisz funkcję `minimum`:

```
int minimum(double* tab, int n)
```

która zwraca indeks najmniejszego elementu n-elementowej tablicy `tab`.

Funkcja main() programu powinna wyglądać w sposób następujący:

```
int main()
{
  double t[10] = {0., 0., 0., 0., 0., 0., 0., 0., 0., 0.};
  drukowanie(t, 10);
  losowanie(t, 10);
  drukowanie(t, 10);
  cout << "indeks minimum: " << minimum(t, 10) << endl;
  return 0;
}
```

Zadanie 55. (Implementacja funkcji swap)

Napisz funkcję, która zamienia miejscami wartości swoich dwóch argumentów. Niewykonalne? – przekaż do funkcji argumenty przy użyciu wskaźników albo referencji.

Zadanie 56. (Funkcja o dwóch wartościach)

Napisz funkcję, która pobiera dwa argumenty i zwraca dwa odrębne wyniki. Jednym z wyników powinien być iloczyn obu argumentów, a drugim ich suma. Ponieważ funkcja może bezpośrednio zwracać tylko jedną wartość, druga powinna być zwracana poprzez parametr wskaźnikowy albo referencję.

Zadanie 57. (Porównywanie liczb *)

Napisz funkcję, która zwraca większą z dwóch podanych zmiennych całkowitych oraz umożliwia nadanie jej nowej wartości. Funkcji powinno dać się użyć następująco:

```
int a=3, b=7; max(a,b)=0;
cut << a << " " << b << endl;
```

Wynikiem działania tego fragmentu programu powinno być wypisanie liczb **3 0**.

Zadanie 58. (Sformatowana tabliczka mnożenia)

Napisz funkcję tworzącą dwuwymiarową tabliczkę mnożenia o dowolnej wielkości. Funkcja nie powinna wyświetlać tabliczki mnożenia, a jedynie ją generować. Następnie napisz drugą funkcję, której zadaniem jest wyświetlenie tabliczki mnożenia, ładnie sformatowanej. Przydziel pamięć potrzebną do stworzenia tablicy operatorem *new*, po skończeniu pracy zwolnij pamięć operatorem *delete*.

Zadanie 59. (Odwrócony Pan Tadeusz)

Zaimplementuj stos zmiennych typu *string*. Korzystając z tej struktury napisz program, który wczytuje ze standardowego wejścia ciąg słów, a następnie wypisuje je na standardowe wyjście w odwrotnej kolejności, oddzielone spacjami. Program przetestuj na tekście *Pana Tadeusza* i *Hamleta*.

Zadanie 60. (Kwestionariusz osobowy)

Napisz funkcję, która prosi użytkownika o podanie w dwóch osobnych zmiennych imienia i nazwiska, a następnie zamienia je miejscami. Funkcja powinna zwracać obie wartości za pośrednictwem dodatkowych parametrów wskaźnikowych (lub referencji) przekazywanych do niej podczas wywołania. *Dodatkowo*: Zmodyfikuj program w taki sposób, aby prosił użytkownika o podanie nazwiska tylko wtedy, gdy w parametrze dotyczącym nazwiska funkcja otrzyma wskaźnik o wartości NULL.

Zadanie 61. (Obliczenia statystyczne 2)

Niech dana będzie próba losowa N wartości x_i. Średnia i odchylenie standardowe z tej próby wynoszą odpowiednio:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \sigma_x = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2}$$

Następnie napisz program, który generuje próbę 1000 liczb pseudolosowych z rozkładu płaskiego w przedziale od 0 do 1, a następnie wypisuje na standardowe wyjście średnią i odchylenie standardowe. Skorzystaj z funkcji o deklaracji:

```
void statistics (double *s, double *m, double tab[], int size)
```

Zadanie 62. (Odwrócony Pan Tadeusz 2 *)

Czasem zamiast pisać samodzielnie potrzebną strukturę danych, warto rozejrzeć się dookoła – prawdopodobnie ktoś wykonał już to zadanie. Korzystając ze struktury ¡stack¿ zawartej w bibliotece STL napisz program, który wczytuje ze standardowego wejścia ciąg słów, a następnie wypisuje je na standardowe wyjście w odwrotnej kolejności, oddzielone spacjami. Program przetestuj na tekście *Pana Tadeusza* i *Hamleta*.

Zadanie 63. (Konwerter systemów liczbowych 2)

Do przedstawienia zadanej liczby naturalne n w systemie pozycyjnym o podstawie m można posłużyć się stosem liczb całkowitych. Algorytm przedstawia się następująco: resztę z dzielenia n przez m odkładamy na stos, a następnie zastępujemy n ilorazem z dzielenia n przez m. Czynności te powtarzamy dopóki n jest niezerowe. Następnie kolejno zdejmujemy liczby ze stosu i wypisujemy od lewej do prawej z tym, że zamiast liczby 10 wypisujemy literę A, i tak dalej.

Napisz program, który wczytuje ze standardowego wejścia liczby n oraz m, a następnie wypisuje na standardowe wyjście liczbę n w systemie pozycyjnym o podstawie m. Program powinien zawierać własną implementację stosu liczb całkowitych.