Zad. 1.

- Załóż katalog `programowanie` w którym będziesz pracować (`mkdir programowanie`) i przejdź do niego (`cd programowanie`)
- (*Ta część jest dla osób nie mających żadnego doświadczenia z C++. Osoby znające C++ proszę o zrobienie pierwszego zadania domowego, dostępnego na platformie Kampus2*) Za pomocą dowolnego edytora (np. Visual Studio) przepisz (nie kopiuj z pliku pdf) program jak poniżej:

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

int main{

  cout << "Podaj liczbe całkowita= " << endl
  int liczba1
  cin >> liczba1;

  cout << Podaj liczbe rzeczywista=  << endl;
  double liczba2;
  cin >> liczba2

  wynik=liczba1/liczba2

  cout << "wynik= " < wynik << endl;

  return 0;
}
```

- Nadaj plikowi nazwę np. prog.cpp
- Skompiluj program za pomocą polecenia g++ (`g++ prog.cpp`)
- Do kompilacji dodaj flagę `-Wall`
- Popraw błędy i uruchom program
- Dodatkowo: sprawdź sam etap preprocesora i kompilacji (patrz - materiał wykładowy). Opcje polecenia `g++` można znaleźć pisząc `man g++`. Jakie pliki powstają we wszystkich trzech przypadkach ?

Zad. 2. Napisz program który:

1. wypisze na ekran:
   Podaj dwie liczby całkowite
2. wczyta te dwie liczby;
3. obliczy ich iloraz, resztę z dzielenia pierwszej liczby przez drugą, średnią arytmetyczną oraz geometryczną;
4. przed próbą dzielenia liczb przez siebie i liczeniem pierwiastka sprawdza poprawność wprowadzonych danych i w przypadku dzielenia przez zero lub pierwiastka z liczby ujemnej przerywa działanie programu (`return 1;`);
5. wypisze na ekran: `a/b =` dd
   `reszta z dzielenia a/b =` rr
   `Średnia arytmetyczna a i b =` xx
   `Średnia geometryczna a i b =` yy

Napisy `dd`, `rr`, `xx` i `yy` muszą być zastąpione wynikami obliczeń.
Do policzenia średniej geometrycznej użyj funkcji `sqrt` z biblioteki `cmath` (konieczne jest dodanie na początku pliku linii `#include<cmath>`).
Operator reszty z dzielenia to %
Przykładowo: 15%4 to reszta z dzielenia 15 przez 4.

Zad. 3. Napisz program znajdujący największą liczbę całkowitą spośród trzech podanych z klawiatury (i wypisujący na ekran jej wartość). Uwaga: spróbuj napisać jak najkrótszy algorytm do znajdowania największej liczby spośród podanych trzech (użyj tylko dwóch instrukcji warunkowych `if` bez `else`, bez && i bez ||).

Zad. 4. Napisz program szukający pierwiastków równania kwadratowego. Program powinien:

1. wypisywać na ekran komunikat:
   `Podaj współczynniki równania kwadratowego`
   a następnie wypisywać:
   `Podaj a`
   i wczytywać z klawiatury `a`.
   Analogicznie dla `b` i `c`.
2. Powinna być tworzona nowa zmienna `delta` i pod nią podstawiona wartość wyliczonej Δ równania kwadratowego. **Wypisz na ekran wartość obliczonej delty**.

3. Następnie sprawdzane powinny być trzy warunki : ($\Delta < 0$, $\Delta = 0$ i $\Delta > 0$) i w zależności od wartości Δ program powinien wypisywać na ekran wartości pierwiastków z komentarzem: `Rownanie nie ma rzeczywistych pierwiastkow`, `Rownanie ma jeden pierwiastek = ...` lub `Rownanie ma dwa pierwiastki: x1 = ...i x2 = ....`

Ważne – sprawdź czy program prawidłowo działa i znajduje pojedynczy pierwiastek dla a=1, b=-0.2 i c=0.01

Zad. 5. Napisz program, który wczytuje ze standardowego wejscia rok, a następnie wypisuje na standardowe wyjście czy ten rok jest przestępny czy nie. Na początku powinno być sprawdzane, czy nie podaliśmy liczby mniejszej lub równej zeru. W przypadku podania błędnych danych program powinien przerywać działanie z odpowiednim komunikatem (użyj polecenia `return`). Rok przestępny spełnia jeden z warunków:

- jest podzielny przez 4 z wyjatkiem lat podzielnych przez 100
- jest podzielny przez 400

Zad. 6. Napisz program, który będzie wczytywał dwie liczby całkowite będące dolną i górną granicą przedziału liczb całkowitych i będzie zliczał i wypisywał na ekran ile liczb parzystych znajduje się pomiędzy wczytanymi liczbami. Po wczytaniu granic program powinien sprawdzać czy dolna granica jest mniejsza od górnej, a jeśli nie to powinien wypisywać komunikat, że jest problem i przerywać działanie za pomocą polecenia:
`return 1;`

Zad. 7. Napisz program sumujący kwadraty kolejnych n liczb naturalnych i wypisujący wynik na ekran. n podawane jest z klawiatury.

Zad. 8. Napisz program do liczenia n! Porównaj dla jak dużych liczb wynik ma sens w przypadku operowania na liczbach całkowitych i zmiennoprzecinkowych.

Zad. 9. Użyj **jednej** pętli `while` do wypisania na ekran obok siebie kolumn dwóch liczb: pierwszej z liczbami od 0 do n i drugiej z liczbami od n do 0. n powinno być wczytywane z klawiatury. Spróbuj stworzyć jak najkrótszy algorytm.

Zad. 10. Napisz program, który będzie prosił o podanie ze standardowego strumienia wejściowego dwóch liczb rzeczywistych i obliczał ich iloraz. Jeżeli liczba przez którą dzielimy jest równa zeru, powinien prosić o podanie innej pary liczb i tak do skutku. Użyj nieskończonej pętli `while` i instrukcji `continue` i `break`.

Zad. 11. Napisz program, który będzie znajdował największy wspólny dzielnik dwóch liczb całkowitych. Znajdź w internecie na czym polega algorytm Euklidesa i go zastosuj.

Zad. 12. Napisz program sumujący wyrazy szeregu $1/10^i$, dla i od 0 do 20. Wypisuj chwilowe wartości sumy, po dodaniu każdego wyrazu. Po każdym wypisaniu na ekran dodaj przejście do nowej linii. W tym samym programie zrób dwie pętle różniące się tylko tym, że raz zmienna przechowywująca sumę będzie typu float, a raz typu double. **Porównaj liczbę cyfr znaczących w obu wypadkach.** Przy wypisywaniu wyników na ekran zwiększ liczbę wypisywanych cyfr do 20.
Liczbę wypisywanych cyfr można zwiększyć w następujący sposób:

```
#include<iostream>
#include<iomanip>
using namespace std;
int main(){
  ...
  cout << setprecision(20) << wynik << endl;
  ...
  return 0;
}
```

Zad. 13. (`Dla zaawansowanych:`)
Napisz program, który rysuje na ekranie kwadrat (prostokąt) o boku n (podawane z klawiatury), z zaznaczonymi obiema przekątnymi. Boki kwadratu i obie przekątne rysowane są dowolnymi znakami wczytywanymi z klawiatury. Polecenie wypisujące na ekran znak powinno pojawić się tylko raz.
Wskazówki:
Docelowo użyj dwóch struktur pętlowych.
Program powinien być pisany etapami:

- jeden wiersz
- pełny kwadrat
- sama ramka
- ramka z jedną przekatną
- ramka z dwiema przekątnymi

Zad. 14. (`Dla zaawansowanych:`)
Napisz program rysujący tabliczkę mnożenia, np. dla n=5 (n podawane z klawiatury):

```
 1    2    3    4    5
 2    4    6    8   10
 3    6    9   12   15
 4    8   12   16   20
 5   10   15   20   25
```

Wskazówki Przy wypisywaniu wyników na ekran ustal stałą szerokość pola, i cyfry przesuń do prawej strony np.: `cout << right << setw(4) << n;`

Zad. 15. (`Dla początkujących:`)
Napisz dwie funkcje: jedną liczącą pole kwadratu o boku `a` i drugą liczacą pole koła o promieniu `r`. Funkcje nie powinny niczego wypisywać na ekran.
Napisz program, który będzie wypisywał na ekran pytanie : `Pole kwadratu czy kola ? (wpisz kwadrat lub kolo)`. Jeżeli użytkownik wpisał `kolo` to program powinien wypisywać pytanie o długość promienia, wczytywać promień i liczyć pole (korzystając ze stworzonej funkcji) i wypisywać pole koła. Jeżeli użytkownik wpisał `kwadrat` to program powinien wypisywać pytanie o długość boku, liczyć pole (korzystajac z funkcji) i wypisywać pole kwadratu. Do wczytywania i przechowywania napisów (nazw figur) użyj zmiennej (obiektu) typu `string`. W przypadku podania błędnych danych program powinien przerywać działanie.

Zad. 16. Napisz funkcję `podzielna`:
`bool podzielna(int liczba1, int liczba2)`
sprawdzajacą czy liczba podana jako pierwszy argument funkcji jest podzielna przez liczbę podaną jako drugi argument funkcji i zwracająca typ logiczny (`bool`). Typ `bool` przyjmuje tylko dwie wartości: `true` lub `false`. Wewnątrz funkcji powinno być sprawdzenie czy `liczba2` jest równa zero. Jeśli jest, to użyj funkcji `exit` z biblioteki `cstdlib` do przerwania programu (`exit(EXIT_FAILURE);`).
Napisz program, który będzie wczytywał liczbę której podzielność chcemy sprawdzić i liczbę która będzie dzielnikiem i przy użyciu stworzonej funkcji wypisywał na standardowe wyjście czy pierwsza z podanych liczb jest podzielna przez drugą. Funkcja nie może niczego wypisywać na ekran.

Zad. 17. Napisz funkcję, która będzie obliczać sumę wszystkich dzielników podanej liczby całkowitej. Napisz program, który używa tej funkcji dla liczb wczytanych ze standardowego strumienia wejściowego i wypisuje sumę jej dzielników na standardowy strumień wyjściowy.

Funkcja nie może niczego wypisywać na ekran. Dla zaawansowanych: zmodyfikuj funkcję tak, aby obliczała także liczbę dzielników.

Zad. 18. Napisz funkcję, która będzie sprawdzać, czy podana liczba jest liczbą doskonałą (suma jej dzielników właściwych jest równa danej liczbie). Np. 6 = 1+2+3
Przy użyciu tej funkcji znajdź i wypisz 3 pierwsze liczby doskonałe.

Zad. 19. Napisz dwie funkcje **o tej samej nazwie**, które zamieniają wartości dwóch zmiennych podanych jako ich argumenty. Jedna funkcja ma zamieniać ze sobą dwie liczby rzeczywiste, a druga dwa znaki (`char`). Użyj obu funkcji w tym samym programie. Wypisz liczby i znaki przed i po zamianie.

Zad. 20. Napisz funkcję, która liczy silnię metodą rekurencyjną (funkcja woła samą siebie). Sprawdź jej działanie w programie. Porównaj swoje rozwiązanie z rozwiązaniem znajdującym się w materiałach wykładowych o funkcjach.

Zad. 21. (`Dla zaawansowanych:`)
Napisz funkcję rekurencyjną, która wczytuje liczbę dziesiętną i wypisuje tę liczbę w systemie dwójkowym. Sprawdź jej działanie w programie.

Zad. 22. Napisz funkcję: `losujint`, która zwraca **całkowite** liczby losowe z przedziału [a,b]. Napisz program, który korzystając z tej funkcji robi symulację:

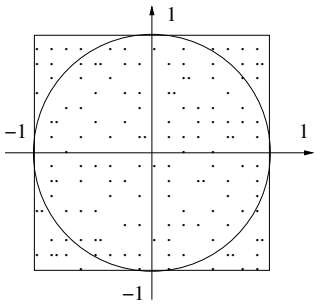
1. rzutu zwykłą kostką;
2. rzutu kostką magiczną, gdzie po rzucie może wypaść dowolna, wielka litera z alfabetu (bez ą,ć,ó ...).

Do zapisywania pojedynczych znaków wykorzystaj typ zmiennej `char` (a tym samym kod ASCII, który przypisuje znakom kolejne liczby całkowite od 0 do 127: `https://pl.wikipedia.org/wiki/ASCII`). Skorzystaj z funkcji `rand` i `srand` z biblioteki `cstdlib`.

Zad. 23. Napisz program do liczenia pola koła wpisanego w kwadrat o boku 2 (rysunek 1), zgodnie z następującym przepisem:

- losujemy `n` punktów (= niezależnie współrzędną x i y, `n` podawane z klawiatury) o współrzędnych z rozkładu płaskiego zawartych w przedziale [-1,1] (napisz funkcję `losujdouble`, która zwraca **rzeczywiste** liczby losowe z przedziału [a,b]);
- dla każdego punktu sprawdzane jest czy punkt leży wewnątrz koła o promieniu 1 i środku (0,0) i zliczane są punkty leżące wewnątrz koła

- obliczany jest stosunek liczby punktów leżących wewnątrz koła do całkowitej liczby punktów i wyliczane pole koła, czyli przybliżenie liczby π. Użyj funkcji `rand()`.



Rysunek 1: Rysunek do zadania

Zad. 24.

Przygotowanie do kartkówki

1. Napisz program liczący sumę szeregu

$$\frac{1}{1\cdot 2}+\frac{1}{2\cdot 3}+\frac{1}{3\cdot 4}+\ldots+\frac{1}{n(n+1)}$$

   Zrób dwie wersje tego programu: z pętlą `for`, gdzie liczba dodawanych wyrazów szeregu jest wczytywana z klawiatury i z petlą `while` z warunkiem, że dodajemy kolejne elementy sumy, dopóki są większe od 10^{-5} (1e-5).

2. Napisz program sprawdzający czy dana liczba `k` jest liczbą pierwszą.

   - Zastosuj algorytm, który dla liczby `k` sprawdza podzielność przez kolejne liczby z przedziału $[2, \sqrt{k}]$ (`i*i<=k`). Sprawdzaną liczbę wczytuj ze standardowego strumienia wejściowego, czyli z klawiatury. Algorytm zapisz w funkcji
     `bool czy_pierwsza(int liczba);`
   - Znajdź n pierwszych liczb pierwszych przy użyciu funkcji z poprzedniego punktu.

3. `Dodatkowo`: Napisz program znajdujący liczby z przedziału 1 do 1e6 metodą sita Erastotenesa `https://pl.wikipedia.org/wiki/Sito_Eratostenesa`

Zad. 25. Napisz program rysujący dywan $N \times M$ zbudowany z losowo wybieranych znaków o kodach ASCII od 32 do 126.

Zad. 26. Napisz program wczytujący ze standardowego strumienia wejściowego pojedyncze słowa (Ctrl-D kończy wczytywanie) i wypisujący liczbę wczytanych słów, liczbę wczytanych znaków, średnią długość słowa i liczbę wystąpień litery a (wielkiej lub małej).

Zad. 27. Napisz program wczytujący słowa ze standardowego strumienia wejściowego i przepisujący je do standardowego strumienia wyjściowego zastepując litery alfabetu łacińskiego literami znajdującymi się w alfabecie o n pozycji dalej (po dojściu do końca alfabetu wracamy do początku). Program powinien działać tak samo dla wielkich i małych liter.

Zad. 28. Stwórz dwie tablice o długości 20. Pierwszą z nich wypełnij kolejnymi potęgami liczby π. Następnie korzystając z pierwszej tablicy, wypełnij drugą tablicę tymi samymi liczbami w odwróconej kolejności. Spróbuj napisać ten program bez używania funkcji `pow`.

Zad. 29. Napisz program, który za pomocą funkcji `getline` wczytuje ze standardowego strumienia wejściowego wyraz zapisany w całości małymi literami i sprawdza czy napis jest palindromem (wyraz brzmi tak samo czytany od lewej do prawej jak od prawej do lewej). Sprawdzanie czy dany napis jest palindromem powinno odbywać się w funkcji: `czy_palindrom`.
Wersja zaawansowana: program powinien sprawdzać czy całe zdanie napisane w całości małymi literami jest palindromem (spacje pomijamy). Przykład: `kobyla ma maly bok` jest palindromem.

(`Dodatkowe, dla zaawansowanych:`)

Wskaźniki i tablice – zadanie na rozgrzewkę:

1. Wskaźniki:
   - stwórz zmienną typu `char` o dowolnej nazwie i wskaźnik pokazujący na miejsce w pamięci gdzie ta zmienna się znajduje. Zmienną zainicjalizuj literą 'K';
   - Używając wskaźnika wypisz wartość zmiennej.
2. Tablice:
   - stwórz tablicę obiektów typu int, o długosci 6;
   - wypełnij ją kwadratami kolejnych liczb naturalnych (skorzystaj ze standardowej pętli `for`, do kolejnych elementów tablicy odwołuj się podając nazwę tablicy i nawias kwadratowy zawierający numer indeksu tablicy);

- ustaw wskaźnik na początku tablicy i wypisz w jednym wierszu zawartość tablicy używając wskaźnika;
- wypisz drugi raz zawartość tablicy w jednym wierszu, korzystając tym razem z uproszczonej postaci pętli `for` (`for` zakresowe, z materiałów wykładowych).

(`Dodatkowe, dla zaawansowanych:`)

Zad. 30. Stwórz kolejną wersję funkcji, która zamienia ze sobą dwie liczby/dwa znaki, a jej argumentami są tym razem dwa wskaźniki. Sprawdź, czy liczby/znaki zostały zamienione ($\rightarrow$ materiały wykładowe).

(`Dodatkowe, dla zaawansowanych:`)

Zad. 31. Napisz funkcję `dlugosc`, która przyjmuje jako argument tablicę znakową (C-string) i zwraca długość napisu w tej tablicy. Użyj pętli `while` i wykorzystaj fakt, że ostatni znak za napisem ma kod Ascii 0.
Napisz funkcję w dwóch wersjach:

```
int dlugosc1(char znaki[]);
```

i

```
int dlugosc2(char *wsk);
```

Te funkcje zdefiniowane w tym samym pliku, nie mogą mieć takiej samej nazwy. Czy wiesz dlaczego ?
Funkcje `dlugosc` umieść w programie **za** funkcją `main`. Napisz program, który tworzy tablicę znakową i inicjalizuje ją literami tworzącymi początek inwokacji "Litwo Ojczyzno moja":

```
char inwokacja[] = "Litwo Ojczyzno moja";
```

a następnie uruchom funkcję dla w/w tablicy.
Dla porównania:

1. użyj gotowej funkcji `strlen` z biblioteki `cstring`;
2. stwórz zmienną typu string, zainicjalizuj ją tym samym ciągiem znaków i korzystając z funkcji length() wypisz długość napisu. Przykład:
   ```
   string zwierzak="Kot";
   cout « zwierzak.length() « endl;
   ```

Wektory – zadania na rozgrzewkę:

1. Pierwsza wersja zadania:
   - Wczytaj długość wektora `n` z klawiatury
   - Stwórz wektor `los1` o długości `n` i wypełnij go liczbami losowymi z przedziału od 0 do RAND_MAX (użyj funkcji `rand()` z biblioteki cstdlib)

- Wypisz na ekran zawartość wektora (w jednym wierszu). Użyj standardowej pętli for i metody `.size()`

2. Druga wersja zadania:

   - Długość wektora `n` wczytaj z klawiatury.

   - Stwórz pusty wektor `los2`. Uzywając funkcji `.push_back` wypełnij go `n` losowymi liczbami, też z przedziału od 0 do RAND_MAX.
   - Używając zakresowej pętli `for` wypisz na ekran zawartość wektora (w jednym wierszu).

Zad. 32. Napisz program, który tworzy w pamięci vector liczb rzeczywistych o długości n (n wczytywane z klawiatury), wypełnia go losowymi liczbami rzeczywistymi z przedziału od zera do jednego (skorzystaj z funkcji `rand()`, `rand()/RAND_MAX`), a następnie oblicza i wypisuje na standardowe wyjście średnią arytmetyczną oraz odchylenie standardowe z tej próby.
Program powinien zawierać cztery funkcje: funkcję wypełniajacą wektor, funkcję wypisującą zawartość wektora na ekran, funkcję liczącą średnią elementów wektora i funkcję liczacą odchylenie standardowe.
W wersji rozszerzonej dodaj funkcję losującą liczby rzeczywiste z dowolnego przedziału od a do b.
Umieść funkcje za funkcją `main` !
Nie korzystaj z gotowych funkcji ze standardowej biblioteki.

Zad. 33. (`Przygotowanie do kartkówki z funkcji, do poćwiczenia w domu:`)
Napisz funkcję, liczącą maksymalną wysokość rzutu ukośnego ze wzoru

$$h_m = \frac{1}{2}\frac{(v_0 \sin(\alpha))^2}{g}$$

gdzie prędkość początkowa v_0, kąt α i przyspieszenie ziemskie g są argumentami funkcji. Napisz drugą funkcję przeliczajacą stopnie na radiany. Napisz program, który używa pierwszej funkcji dla v_0, α i g wczytanych ze standardowego strumienia wejściowego i wypisuje obliczoną wartość h_m na standardowy strumień wyjściowy. Program powinien prosić o podanie v_0 w jednostkach $[m/s]$, a g w jednostkach $[m/s^2]$. Przed wczytaniem wartości kąta, program powinien wypisywać pytanie `Kąt podajesz w stopniach czy w radianach ? - Wpisz stopnie lub radiany`, a następnie `Podaj wartość kąta` i w zależności od tego co wpisał użytkownik przeliczać (przy użyciu drugiej funkcji) bądź nie podany kąt na radiany. Program powinien sprawdzać, czy v_0, α i g są większe lub równe zero i w razie podania złych danych przerywać działanie ze stosownym komunikatem.

Zad. 34. (`Przygotowanie do kartkówki z funkcji, do poćwiczenia w domu:`)
Napisz funkcję `expTaylor`, która dla zadanych argumentów `double x` i `int kmax` zwraca przybliżenie funkcji

$$\mathrm{e}^x \approx \sum_{n=0}^{k_{max}} \frac{x^n}{n!}$$

Wyrażenie `n!` wyznaczaj w odrębnej funkcji.
W funkcji `main()` poproś użytkownika o podanie `x` oraz `kmax`, a następnie wypisz: wartość przybliżenia, wartość funkcji exp(x) z biblioteki <cmath> oraz różnicę między nimi. Na koniec zastanów się, czy musimy liczyć coraz większe silnie i potęgi? Podziel wyraz a_{n+1} przez a_n, wyciągnij wnioski i adekwatnie zmodyfikuj swoją funkcję.

Zad. 35.

- Napisz program, który wypełni wektor o długości `n` całkowitymi liczbami losowymi z przedziału od 0 do 1000 i znajdzie największą z nich.
- W wersji podstawowej `n` wczytaj z klawiatury (ze standardowego strumienia wejściowego)
  - W wersji zaawansowanej liczba `n` powinna być przekazywana do wnętrza programu jako argument wywołania programu. Zabezpiecz program przed uruchomieniem go bez podania wartości n.
    Wersja 1 wywołania programu dla n=20:
    `./a.out 20`
    Wersja 2 wywołania programu:
    `./a.out -n 20`
- Podziel program na nastepujace funkcje:
  - do znajdowania indeksu elementu wektora (przy użyciu własnego algorytmu) w którym znajduje się największa liczba;
  - do wypełniania wektora;
  - do wypisywania jego zawartości na ekran;
  - do wczytywania argumentów wywołania programu (parser) (ta funkcja tylko w wersji dla zaawansowanych).
- Użyj funkcji `rand()` z biblioteki `cstdlib` (`rand()%1001`).
- Na początku programu użyj funkcji `srand(time(0))` żeby za każdym razem wypisywały się inne liczby.

Dodatkowo: Porównaj swój wynik z wynikiem działania standardowego algorytmu `max_element` z biblioteki `algorithm`
`cout « *max_element(vec.begin(),vec.end()) « endl;`

Zad. 36. Napisz program wypełniający vector o długości `n` (`n` wpisywane z klawiatury) losowymi liczbami z przedziału [0,1), a następnie sortujący liczby w wektorze metodą bąbelkową.
Przykładowy podział na funkcje:

- Funkcja zwracajaca liczbę losową z przedziału [0,1)
  `double losuj()`
- Funkcja zamieniajaca miejscami dwie liczby
  `void zamiana(double & pozycja1, double & pozycja2)`
- Funkcja sortująca metodą bąbelkową (i ustawiająca elementy w wektorze od najmniejszego elementu do największego)
  `void bubble_sort(vector<double> &tab)`
  - Zewnętrzna pętla `do ... while{zamienione}` - powtarza się jeśli w wektorze któreś z elementów zostały zamienione ($\Rightarrow$ może jeszcze coś trzeba będzie zamienić. Jesli już nic nie było zamieniane, to znaczy że wektor jest uporządkowany.)
  - Wewnętrzna pętla przechodzi przez wszystkie elementy wektora i sprawdza czy *i-ty* element jest większy od elementu *i+1*. Jeśli tak, to używa funkcji `zamiana` i ustawia zmienną logiczną `zamienione` na `true`.
  - W funkcji `main` wypisywane są wszystkie elementy wektora przed i po sortowaniu i wywoływana jest funkcja `bubble_sort`.

Przykład działania algorytmu sortującego:

$$\underline{43}52 \rightarrow 3\underline{45}2 \rightarrow 34\underline{52} \rightarrow 3425$$

$$\underline{34}25 \rightarrow 3\underline{42}5 \rightarrow 32\underline{45}$$

$$\underline{32}45 \rightarrow 2\underline{34}5 \rightarrow 23\underline{45}$$

Zad. 37. (`Dla zaawansowanych:`)
Napisz program `grafika`, który tworzy vector STL, którego elementami są vector'y STL (`vector<vector<int> > v;`) Uwaga ! W nie najnowszych standardach języka C++ spacja za `<int>` jest ważna. Pojemnik powinien

zostać wypełniony zerami i jedynkami we wzór szachownicy. Program powinien zapisywać do standardowego strumienia wyjściowego zawartość wektora wektorów w formie tablicy dwuwymiarowej, poprzedzoną następującymi linijkami:

```
P1
nk nw
```

gdzie `nk` i `nw` powinny zostać zastąpione odpowiednio liczbą kolumn (rozmiar `vector<int>`) i liczbą wierszy (rozmiar `vector<vector<int> >`). Rysunek szachownicy zapisany w ten sposób jest grafiką w formacie pbm, w którym zera reprezentują białe piksele, a jedynki czarne.
Pierwsza linia `P1` oznacza format grafiki (pbm). Druga linia to liczba kolumn i wierszy.
Program powinien:

- wczytywać liczbę kolumn `nk` i liczbę wierszy `nw`
- w wersji zaawansowanej wypełnianie wektora wektorów powinno się odbywać w funkcji o prototypie:
  ```
  void wypelnij (vector<vector<int> > &szach, int nw, int nk);
  ```

Zapisz obrazek, przekierowując standardowy strumień wyjściowy do pliku `szachy.pbm` w następujący sposób:
`./grafika > szachy.pbm`
Obejrzyj stworzony rysunek dowolnym programem do oglądania grafik (np. `gimp`).

Klasy

Zad. 38. Napisz klasę Trojkat, która będzie reprezentować trójkąt. Stwórz prywatne składowe tej klasy - boki trójkąta. Stwórz funkcje składowe klasy:

- do ustawiania długości boków trójkąta (trzy funkcje składowe typu `set`);
- trzy funkcje składowe typu `get` do odczytywania wartości poszczególnych boków;
- do liczenia pola trójkata z wzoru Herona;
- do liczenia obwodu trójkąta;
- do sprawdzania, czy podane boki w ogóle tworzą trójkąt.

- do wczytywania trzech długości boków ze standardowego strumienia wejściowego.

Stwórz obiekt klasy Trojkat i użyj wszystkich jego funkcji.

Zad. 39. Do klasy Trojkat dodaj:

- konstruktor standardowy w którym wszystkie długości boków ustawiane są jako zerowe
- konstruktor z trzema argumentami (trzy długości boków)
- destruktor, który wypisuje tekst, np: `obiekt został zniszczony`

Użyj obu konstruktorów.

Zad. 40. Kolejne modyfikacje klasy Trojkat:

- w konstruktorze niestandardowym i innych miejscach gdzie podawane są długości boków zawołaj funkcję sprawdzającą czy podane boki tworzą trójkąt, a jeśli nie, to przerywaj działanie programu (użyj funkcji `exit` z biblioteki `cstdlib`).
- w funkcji wczytującej długości boków trójkąta dodaj wczytywanie w pętli do momentu, aż użytkownik nie wczyta dodatnich długości boków;

Zad. 41. Podziel program na trzy pliki: `Trojkat.h` (definicja klasy Trojkat), `Trojkat.cpp` (definicje większych funkcji) i `main.cpp` (główny program).

Od tego momentu, w kolejnych zadaniach wszystkie klasy zapisuj w oddzielnych plikach: nagłówkowym `.h` i `.cpp`

Zad. 42. Stwórz prostą wersję klasy o nazwie Zespolona do reprezentowania liczb zespolonych. Klasa powinna mieć prywatne dane składowe, konstruktory, funkcje typu set i get, funkcję liczącą moduł liczby zespolonej, wypisującą daną liczbę.

Zad. 43. Napisz klasę Student. Stwórz prywatne składowe tej klasy: imię, nazwisko, rok studiów, grupa, vector z ocenami
Stwórz funkcje składowe klasy:

- konstruktory (argumentami niestandardowego konstruktora powinny być zmienne reprezentujące: imię, nazwisko, rok studiów, numer grupy);
- funkcję `wczytaj_oceny` do wczytywania z klawiatury do pojemnika vector ocen z Analizy, Algebry, Wstępu do Fizyki, Programowania i Pracowni. W wersji podstawowej pojemnik najpierw powinien być pusty i rozbudowywany w miarę dodawania ocen;

- `print` do wypisywania informacji o studencie;
- funkcję do obliczania średniej ocen;
- funkcje do ustawiania i odczytywania wartości danych składowych klasy (typu `set` i `get`).

W głównym programie stwórz obiekt klasy Student reprezentujący Jana Kowalskiego z 1 roku, który zapisał się do grupy nr 8. Wczytaj jego oceny i wypisz średnią ocen. Przetestuj funkcję `print`.

Po przetestowaniu powyższej wersji zrób drugą, gdzie na liście inicjalizacyjnej konstruktora tworzony jest wypełniony zerami `vector` (z ocenami) o długości 5.

Przygotowanie do kartkówki - pojemnik vector

Zad. 44. Napisz program, który ze standardowego strumienia wejściowego wczytuje w pętli ciąg wyrazów aż do momentu gdy strumień znajdzie się w złym stanie ($\rightarrow$ wykład 5, przykłady ze strony 8 i 7).

- Policz i wypisz na ekran ile zostało wczytanych wyrazów i znaków.
- Znajdź ile razy w tekście występują kolejne litery z alfabetu (bez polskich znaków). W tym celu:
  - Stwórz pojemnik vector o długości takiej ile jest liter w alfabecie w tablicy kodów ASCII.
  - Komórki tego pojemnika powinny docelowo zawierać liczby wystąpień w tekście kolejnych liter (dużych lub małych, bez polskich znaków).
  - Wypisz jakiej literze odpowiada dana komórka i ile razy dana litera wystąpiła w badanym tekście.

Przetestuj swój program na tekście z pliku `shakespeare.txt` (do ściągnięcia z przestrzeni ćwiczeniowej na Kampus2):
`./a.out < shakespeare.txt`.

Zad. 45. Zmieniamy klasę Trojkat tak, aby trójkąt był konstruowany z trzech punktów. W tym celu:

- stwórz klasę Punkt o danych składowych `x` i `y`, będących współrzędnymi punktu; klasę zapisz w plikach Punkt.h i Punkt.cpp

- w klasie Punkt stwórz metodę `print` wyświetlającą współrzędne punktu w formacie `(x,y)`
- w klasie Punkt stwórz również metodę `double distance (const Punkt & p2) const`, która będzie zwracać odległość pomiędzy danym punktem, a punktem p2
- zmień podstawowe dane składowe klasy `Trojkat`, na trzy obiekty klasy `Punkt` (długości boków teraz będą pomocnicznymi danymi składowymi).
- dodaj funkcje typu `get` i `set` w obu klasach, dla wszystkich danych składowych
- zmień konstruktor klasy `Trojkat` tak, aby konstruował trójkąt z trzech obiektów klasy `Punkt`
- zmień konstruktor standardowy klasy `Trojkat` tak, aby konstruował trójkąt z trzech punktów o współrzędnych (0,0)
- zmień funkcję `ustaw` tak, aby konstruowała trójkąt z trzech obiektów `Punkt`
- W konstruktorach i funkcji ustaw wyliczaj długości boków (długości boków pozostają danymi składowymi klasy) . Do obliczania długości boku użyj funkcji `distance`) z klasy `Punkt`.

W funkcji `main` stwórz punkty p1(0,4), p2(3,0) i p3(0,0). Stwórz z nich trójkąt. Wypisz jaki jest jego obwód i pole.

Przerywnik - zapisywanie do pliku i czytanie z pliku

Zad. 46. Napisz program losujący n liczb całkowitych z przedziału [0,10000] (n podawane z klawiatury) i zapisujący liczby parzyste do pliku parzyste.txt, a nieparzyste do pliku nieparzyste.txt
Uwaga ! Nie przechowuj wylosowanych liczb w żadnym pojemniku typu vector, tylko losuj i od razu zapisuj liczby do plików.

Zad. 47. W pliku `trojki.txt` znajdują sie trzy kolumny liczb (plik do ściągnięcia ze strony ćwiczeniowej). Napisz program wczytujący do wektora tylko liczby z drugiej kolumny i wypisujący ile liczb zostało wczytanych. W tym celu:

- zadeklaruj pusty wektor
- w pętli `while` wczytuj trzy liczby do trzech zwykłych zmiennych

- liczby z drugiej kolumny dodawaj do wektora
- za pętlą wypisz jaka jest długość wektora = ile liczb zostało wczytanych

Zastanów się czy wiesz jak policzyć średnią z liczb z dowolnej kolumny bez zapisywania liczb do vector'a (ani żadnego innego pojemnika) ?

Klasy cd ...

Zad. 48. Dalsza rozbudowa klasy Trojkat.

- Dodaj konstruktor kopiujący `Trojkat(const Trojkat & source);` i wpisz w jego środku komunikat: Konstruktor kopiujący
- W głównym programie stwórz obiekt klasy Trojkat, a następnie jego kopie:

```
Trojkat t1(p1,p2,p3);
Trojkat t2(t1);
Trojkat t3=t1;
```

- Stwórz pusty wektor `v` i dodaj do niego trzy obiekty klasy Trojkat.

Zad. 49. Do klasy Trojkat dopisz metodę `void zapisz()` do zapisywania do pliku `trojkat.txt`, w trzech wierszach, współrzędnych trzech wierzchołków trójkąta (lub trzech boków trójkąta). Przetestuj działanie tej funkcji. W kolejnym kroku nazwa pliku powinna być wczytywana z klawiatury w funkcji main i przekazywana jako argument do funkcji `zapisz`.

Zad. 50. Do klasy Trojkat dopisz metodę `void wczytaj(ifstream & nazwa_pliku)` do wczytywania z pliku współrzędnych trzech wierzchołków trójkąta (lub trzech boków trójkąta). Przetestuj działanie tej funkcji.

Zad. 51. W pliku `Krosno.txt` znajdują się wybrane dane meteorologiczne z roku 2006 pochodzące ze stacji IMGW 349210670 w Krośnie. Kolejne wiersze odpowiadają kolejnym dniom roku. W pliku `klucz.txt` opisane są zawartości poszczególnych kolumn.

Stwórz program, który prosi o podanie nazwy wejściowego pliku, a następnie wczyta z pliku zawartości potrzebnych kolumn do jednowymiarowych pojemników typu `vector`. Następnie znajdzie odpowiedź na poniższe pytania i wypisze na ekran uzyskane wyniki. W celu poprawy przejrzystości kodu zaprojektuj i wykorzystaj funkcje pomocnicze, przyjmujące co najmniej jeden argument (`vector` z danymi).

W przypadku braku pliku (zły stan strumienia zaraz po próbie otwarcia pliku) program powinien przerywać działanie.

Pytania:

- Jaka była średnia temperatura w roku, średnie usłonecznienie i średni czas trwania opadu deszczu ?
- W którym dniu w roku był najdłuższy czas trwania opadu deszczu ? Ile to było godzin ?
- Przez ile dni w roku maksymalna temperatura zawierała się w przedziale domkniętym pomiędzy 20 a 25 stopni ? W jakich to było miesiącach ?
  Wskazówka: stwórz dodatkowy 12-elementowy pojemnik vector do zliczania dni w poszczególnych miesiącach.

Przygotowania do ...

Zad. 52. Napisz program, który będzie zawierał definicje i używał trzech funkcji:

1. Funkcja nr 1 – zwracająca sumę cyfr liczby całkowitej.
2. Funkcja nr 2 – odwracająca zawartość pojemnika vector (ostatni element jest pierwszy, przedostatni drugi itd.). Funkcja nie powinna zmieniać oryginalnego vector'a, tylko zwracać nowy vector z inną zawartością.
3. Funkcja nr 3 – do zapisu liczby całkowitej w postaci binarnej, zwracająca `vector` przechowujący liczbę zapisaną w dobrej kolejności (najstarszy bit, odpowiadający najwyższej potędze dwójki z lewej strony).

- Wskazówka: użyj operatora reszty z dzielenia (przez 2 lub 10, w zależności od funkcji)
- Znajdź funkcję z biblioteki standardowej (`https://cplusplus.com/reference/`, plik nagłówkowy `algorithm`, także fragment wykładu o iteratorach) odwracającą zawartość pojemnika `vector` i porównaj działanie tej funkcji z działaniem własnej funkcji.
- Użyj ostatniej funkcji do zamiany liczb z pierwszej kolumny pliku `trojki.txt` na liczby w postaci binarnej i zapisz je w nowej formie do pliku `pierwsza.txt`.

Zad. 53. Dla klasy Punkt dodaj przeładowany `operator«` na wzór tego z klasy `Ulamek` opisanego w materiale wykładowym. Sprawdź działanie tego operatora przy zapisie współrzędnych punktu na ekran i do pliku.

```
Punkt p;
...
cout « p « endl;
plik « p « endl;
...
```

Analogicznie dodaj i przetestuj kolejny przeładowany operator: `operator»`

Zad. 54. Do klasy reprezentującej liczby zespolone dodaj operatory dodawania, odejmowania, dzielenia i mnożenia liczb zespolonych. Jeśli nie masz tej klasy to napisz ją w wersji minimalnej.