https://kuratorium.krakow.pl/wp-content/uploads/2019/03/kti_sp_et_woj_zadania.pdf

KURATORIUM OŚWIATY W KRAKOWIE

INFORMATYCZNY KONKURS TEMATYCZNY DLA UCZNIÓW SZKÓŁ PODSTAWOWYCH
„OD ALGORYTMU DO PROGRAMU" - ETAP WOJEWÓDZKI

MSZANA DOLNA

ZADANIE 2. (0 – 45)

Plik z przykładowymi danymi wejściowymi: **bin.in**
Plik wyjściowy: **bin.out**
Dodatkowe pliki do testowania rozwiązań z większą liczbą danych: **bin100000.in, bin2000000.in**
Do oceny należy oddać plik, zawierający w nazwie *Kod Ucznia*
(np. `K01_zad2.cpp lub K01_zad2.c`)

Opis organizacji danych w pliku wejściowym
W pierwszym wierszu pliku **bin.in** jest zapisana jedna liczba naturalna **n** mówiąca o ilości liczb binarnych zapisanych w kolejnych wierszach pliku. Liczba **n** mieści się w przedziale domkniętym od **2** do **2000000**. W każdym z kolejnych **n** wierszy zapisana jest jedna liczba binarna składająca się maksymalnie z **30** cyfr i rozpoczynająca się zawsze cyfrą „1".

Jakie zmienne trzeba użyć

Typy danych

Skoro już wiemy co to jest zmienna, pora zapoznać się z podstawowymi typami danych jakie są dostępne w C++.

Nazwa typu	Ilość Bajtów	Zakres wartości
bool	1	**false** lub **true**
char	1	od -128 do 127
unsigned char	1	od 0 do 255
wchar_t	2	od 0 do 65'535
short	2	od -32'768 do 32'767
unsigned short	2	od 0 do 65'535
int	4	od -2'147'483'648 do 2'147'483'647
unsigned int	4	od 0 do 4'294'967'295
long	4	od -2'147'483'648 do 2'147'483'647
unsigned long	4	od 0 do 4'294'967'295
long long	8	od -9'223'372'036'854'775'808 do 9'223'372'036'854'775'807
unsigned long long	8	od 0 do 18'446'744'073'709'551'615
float	4	3.4E +/- 38 (7 cyfr)
double	8	1.7E +/- 308 (15 cyfr)
long double	8	1.7E +/- 308 (15 cyfr)

Przykład organizacji danych pliku *bin.in*:

```
5
11
10
10110
10001
10
```

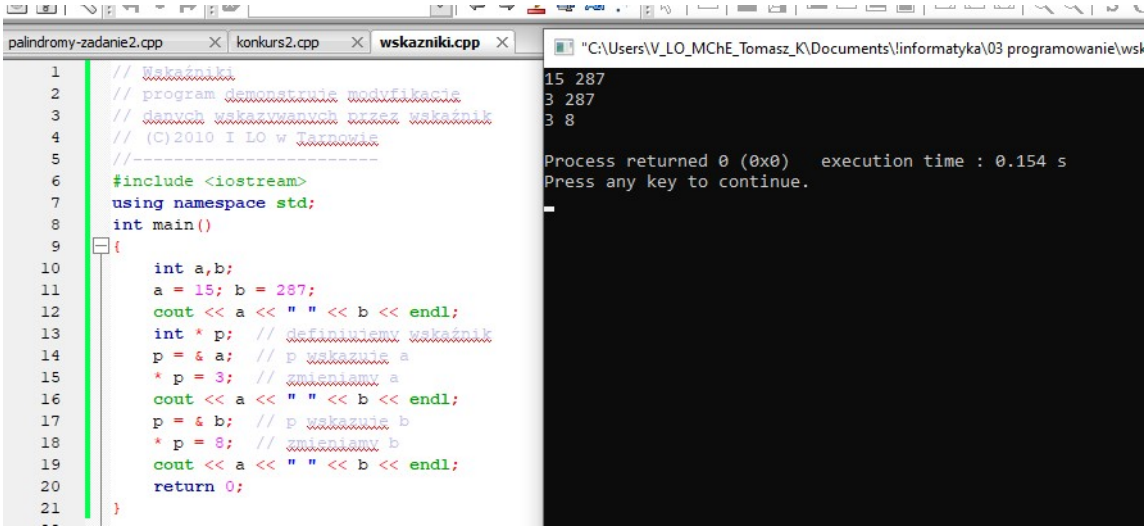
Korzystając z powyższych informacji napisz program, który wykona czynności wymienione w punktach od A do E *(przy każdym z nich podano maksymalną punktację możliwą do uzyskania)*:

A. (0 - 10)
Wczytaj liczby binarne z pliku wejściowego do odpowiednich struktur danych oraz umieść **w kodzie źródłowym w postaci komentarzy** nazwy zmiennych wraz z typami danych jakie zostały użyte do zapamiętania powyższych informacji. Wyświetl na ekranie i zapisz w pliku wyjściowym **ilość liczb binarnych identycznych** do ostatniej wczytanej liczby z wejścia.

Plik z kodem na dysku moim skonkurs.cpp

```
//Zadanie A.
ifstream file_in;
file_in.open("bin.in");
int n;
file_in >> n;
// tablica dynamiczna obiektów klasy string z danymi
string * binarne = new string[n];
for(int i=0;i<n;i++){
    file_in >> binarne[i];
}
file_in.close();//zamknięcia pliku wejściowego
//zapis do pliku file_out
ofstream file_out;
file_out.open("bin.out");
string ostatnia_liczba = binarne[n-1]; //ostatnia liczba w tablicy z danymi
int temp_ostatnia = 0; // zmienna licząca ilość powtórzeń ostatniej liczby
for(int i=0;i<n-1;i++){ //sprawdzenie czy która z liczb w tabeli jest identyczna co ostatnia
    if (ostatnia_liczba==binarne[i])
        temp_ostatnia++;
}
cout << "A\n"<< temp_ostatnia << endl <<endl;
file_out << "A\n"<< temp_ostatnia << endl <<endl; //zapis do pliku
```

Informacje o wskaźnikach są dobrze opisane na stronie:
https://eduinf.waw.pl/inf/utils/010_2010/0513.php



Dane dynamiczne

W pamięci komputera możemy rezerwować obszary na przechowywanie danych określonego typu. Do tego celu służy operator **new**, który stosujemy w sposób następujący:

wskaźnik = **new** *typ*;

new - tworzy obszar pamięci i zwraca jego adres, który zostaje umieszczony we wskaźniku
typ - określa rodzaj informacji, która będzie przechowywana w utworzonym przez **new** obszarze. Typ ten musi być zgodny z typem obiektów wskazywanych przez wskaźnik.

Po wykorzystaniu obszar pamięci można zwrócić do puli systemu za pomocą polecenia:

delete wskaźnik;

Wskaźnik w powyższym poleceniu musi zawierać adres obszaru utworzonego przez **new**. Zwrócona pamięć może zostać wykorzystana do innych celów.

```
1  // Wskaźniki
2  // program demonstruje dana dynamiczna
3  // (C)2010 I LO w Tarnowie
4  //-------------------------
5  #include <iostream>
6  using namespace std;
7  int main()
8  {
9      double * p;      // definiujemy wskaźnik
10     p = new double; // przydzielamy pamięć
11     * p = 3.1415;   // w przydzielonej pamięci umieszczamy dane
12     cout << * p << endl;
13     delete p;       // zwalniamy przydzielony obszar pamięci
14     return 0;
15 }
16
```

```
"C:\Users\V_LO_MChE_Tomasz_K\Documents\!informatyka\03 programowanie\wskaznikiDy
3.1415

Process returned 0 (0x0)   execution time : 0.166 s
Press any key to continue.
```

Tablice dynamiczne

Tablica dynamiczna *(ang. dynamic array)* jest tworzona w czasie uruchomienia programu. Jej rozmiar może być wyliczany. Co więcej, gdy przestanie być potrzebna możemy ją usunąć z pamięci. Dzięki tym własnościom program efektywniej wykorzystuje zasoby pamięciowe komputera.

Tworzenie i wykorzystanie tablicy dynamicznej

1. Definiujemy zmienną wskaźnikową, która będzie przechowywała adres pierwszego elementu tablicy. Jest to zwykła definicja wskaźnika:

   typ * wskaźnik;

2. Przydzielamy obszar pamięci dla tablicy. Stosujemy następującą konstrukcję:

   wskaźnik = new typ[liczba_elementów];

   liczba_elementów określa rozmiar tablicy. Może być to dowolne wyrażenie arytmetyczne.

3. Do elementów tak utworzonej tablicy odwołujemy się poprzez ich indeksy:

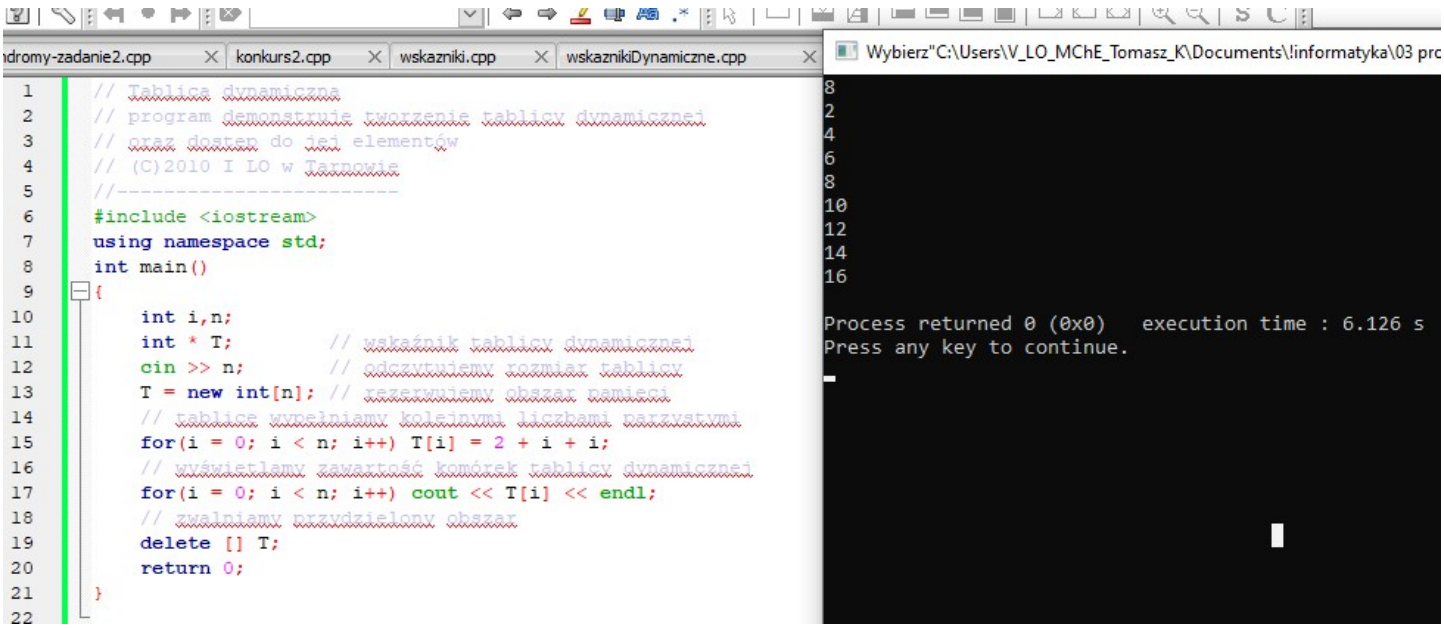
   wskaźnik[indeks]

   indeks - powinien być w zakresie od 0 do liczba_elementów - 1. Kompilator nie sprawdza, czy element o danym indeksie znajduje się faktycznie w tablicy. Należy zachować ostrożność.

4. Gdy tablica dynamiczna przestanie być potrzebna, usuwamy ją z pamięci za pomocą instrukcji:

   delete [] wskaźnik;

Po tej operacji obszar pamięci zajęty przez tablicę zostaje zwrócony do systemu. Wskaźnik można wykorzystać ponownie do innej tablicy wg powyższych punktów.



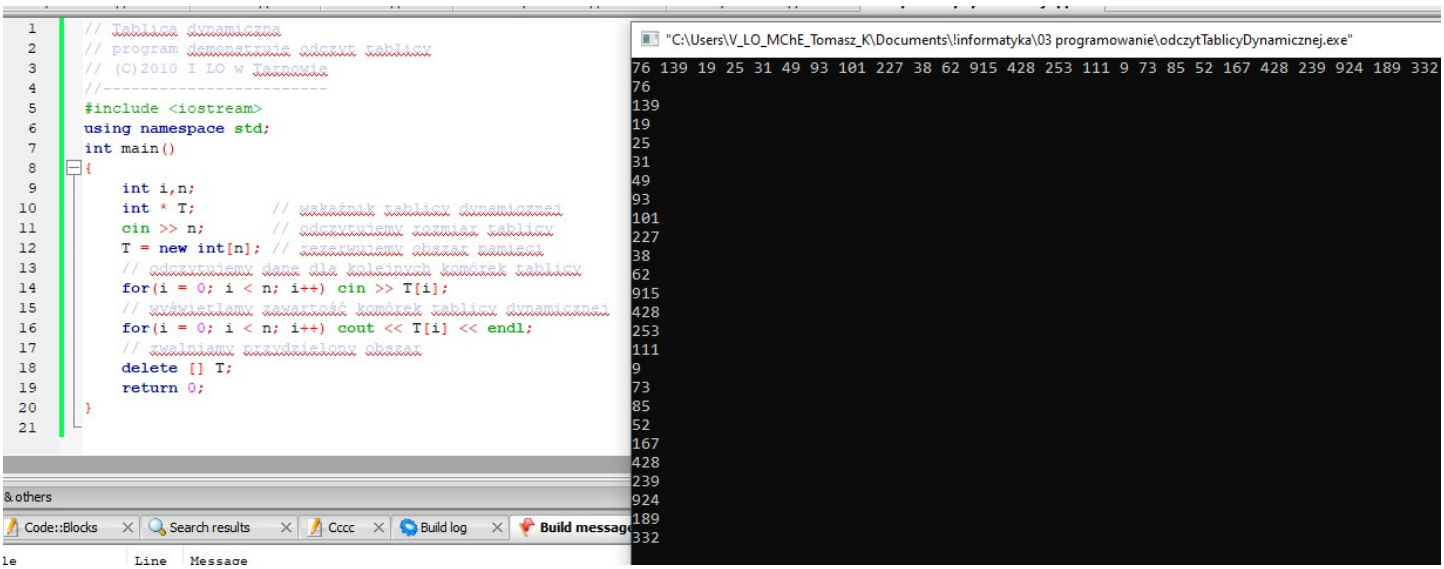
Odczyt tablicy dynamicznej ze standardowego wejścia

Tablicę dynamiczną można tworzyć z danych odczytanych ze standardowego wejścia. W tym celu dane te powinny posiadać określoną postać. Umówmy się, iż pierwsza liczba *n* będzie określała liczbę komórek tablicy dynamicznej, a następne n liczb będzie zawartością tych komórek.

Przykładowe dane:

25
76 139 19 25 31 49 93 101 227 38 62 915 428 253 111 9 73 85 52 167 428 239 924 189 332

Skopiuj powyższe dane do schowka Windows. Następnie uruchom poniższy program:



Po uruchomieniu programu klikamy prawym przyciskiem myszki w pasek tytułowy okna konsoli. Z menu wybieramy opcję Edytuj → Wklej (być może trzeba będzie jeszcze wcisnąć klawisz Enter). Dane ze schowka zostaną wklejone do okna konsoli, program je odczyta i przetworzy w tablicy dynamicznej. Ten sposób pozwala nam szybko i wielokrotnie wprowadzać większą porcję danych do programu bez konieczności ich wpisywania z klawiatury.

Jeśli danych jest bardzo dużo, to lepszym rozwiązaniem będzie przekierowanie standardowego wejścia. Wyszukaj na dysku katalog, w którym zapisany jest plik programu. Katalog ten będzie w katalogu projektowym pod nazwą bin/Debug - dla wersji uruchomieniowej lub bin/Release - dla wersji ostatecznej. Zapisz w tym katalogu plik z danymi, np. pod nazwą d.txt.

Uruchom okno poleceń Windows (w menu Start wybierz opcję Uruchom i wpisz cmd). Za pomocą poleceń cd przejdź do katalogu z programem. Następnie wpisz:

nazwa_programu **< d.txt**

Program będzie odczytywał dane z pliku d.txt zamiast z okna konsoli. Standardowe wyjście również można przekierować do pliku:

nazwa_programu > out.txt

Teraz program zamiast pisać dane do okna konsoli, umieści je w pliku out.txt. Plik ten można odczytać np. za pomocą notatnika Windows i spokojnie przejrzeć sobie treść - szczególnie przydatne, gdy program produkuje dużo danych i nie mieszczą się one w oknie konsoli.

Można jednocześnie przekierować i wejście, i wyjście danych:

nazwa_programu **< d.txt > out.txt**

Teraz program odczyta dane z pliku d.txt, a wyniki zapisze w pliku out.txt. Zwróć uwagę, iż w samym programie nie musisz robić