**Dział programu - Implementacja klasycznych algorytmów iteracyjnych**

**Znajdowanie najmniejszego lub największego elementu w zbiorze**

Napiszemy program, który z k wczytanych liczb wyznacza liczbę najmniejszą. Program wczyta najpierw liczbę k oznaczającą ilość liczb w badanym ciągu. Następnie wczyta k liczb i wyznaczy najmniejszą z nich.

Komentarz:

Najmniejsza wartość będzie zapamiętana w zmiennej **min**. Po wprowadzeniu pierwszej liczby z ciągu przyjmiemy ją za tymczasowe minimum ( niezależnie od wartości!). Po wprowadzeniu każdej następnej liczby sprawdzamy, czy jest ona mniejsza od dotychczas przyjętego minimum. Jeśli tak to wartość tej liczby przypisujemy zmiennej min (nadpisujemy na poprzedniej wartości), jeśli nie jest mniejsza to nic się nie dzieje – sprawdzamy następną. Po wprowadzeniu ostatniej liczby (po przeprowadzeniu k porównań, możemy wyświetlić wartość, jaką przechowuje zmienna min. Oczywiście musimy mieć zmienną licznik, którego wartość to ilość liczb w ciągu który chcemy przeszukiwać. Na początku programu licznik przyjmie wartość ilości badanych liczb (k), a po każdej wprowadzonej liczbie jego wartość zmniejszymy o 1 (k=k-1).

Algorytm w postaci listy kroków:

1. Wczytaj k.
2. Wczytaj a.
3. Zmiennej min przypisz wartość a.
4. Jeśli k jest równe 1, wypisz min i zakończ. //blok decyzyjny, instrukcja if else
5. Wczytaj (następne) a.
6. Zmniejsz k o jeden.
7. Jeśli a<min, przejdź do kroku 3 //druga instrukcja if else, drugi blok decyzyjny
8. Przejdź do kroku 4.

Schemat blokowy algorytmu – wykonany w programie Magiczne bloczki

W tym programie piszemy w tzw. Pseudokodzie, jego słowa kluczowe różnią się od słów w C++, są bliższe językowi Pascal, ale są to pojedyncze słowa angielskie, więc nie będzie problemu z odczytaniem. W C+ znak przypisania to =, natomiast tu będzie :=, tak jak w Pascalu, który był językiem niższego poziomu niż C++.



Schemat blokowy znajdowania min w ciągu liczb

Program w C+ realizujący ten algorytm:



W analogiczny sposób możemy znaleźć element największy w ciągu liczb – należy dokonać zmian:

Linia 6 – int k, a, max

Linia 11 – max=a

Linia 17 – if(a>max)

Linia 18 max=a

No i oczywiście z miana w linii 20.

**Zadanie domowe napisz kod programu, który w ciągu k liczb znajdzie element min i max jednocześnie. Ilość liczb podaje użytkownik. Kod programu o nazwie minmax.cpp należy przysłać do 30 maja.**

**Liczba pierwsza czy złożona – algorytm testujący**

Zacznijmy od przypomnienia definicji liczby pierwszej: **liczba pierwsza to taka liczba naturalna, która ma tylko dwa dzielniki: liczbę jeden i sama siebie.**

Liczby pierwsze stanowią zbiór nieskończony. Są nimi 13 i 7, a nie jest nią liczba 14, bo oprócz tego, że dzieli się przez 14 i 1, to także przez 2 i 7. **Liczba, która nie jest liczbą pierwszą, nazywa się liczbą złożoną. Każda liczba złożona jest iloczynem liczb pierwszych: 14=2\*7.**

**Liczby 0 i 1 nie są ani liczbami pierwszymi, ani złożonymi.**

Więcej na temat liczb pierwszych przeczytasz na stronie projektu GIMPS <http://www.mersenne.org> oraz na stronie <https://miroslawzelent.pl/nauka/> - polecam!

Zastosowanie: duże liczby pierwsze służą do szyfrowania danych. Są też wykorzystywane do tworzenia kodów korekcyjnych do wyszukiwania błędów w przekazie danych, na przykład z satelity.

Napiszemy program, który udzieli odpowiedzi, czy podana na wejściu liczba n jest liczbą pierwszą

Aby to zbadać, będziemy liczbę n dzielić przez kolejne liczby naturalne mniejsze od n, począwszy od liczby 2, i sprawdzać **resztę z dzielenia**. Dlaczego? Jeśli znajdziemy choćby jeden dzielnik, dla którego reszta z dzielenia wynosi 0(zero), to liczba n jest liczbą złożoną. Nie musimy sprawdzać wszystkich dzielników mniejszych od n. Jeśli liczba nie dzieli się przez żadną liczbę całkowitą mniejszą lub równą pierwiastek z n, to jest to liczba pierwsza.

Komentarz:

Dowód tego twierdzenia jest dosyć prosty. Jeżeli liczba n jest liczbą złożoną, to co najmniej jeden dzielnik jest nie większy od $\sqrt{n}$, ponieważ jeżeli jest większy, wówczas n byłoby równe iloczynowi ab, gdzie a>$\sqrt{n}$ i b>$\sqrt{n}$, a zatem a\*b byłoby większe od $\sqrt{n}\* \sqrt{n}$, czyli większe od n, co jest sprzeczne z założeniem.

Algorytm w postaci listy kroków:

1. Wczytaj n.
2. Zmiennej pomocniczej i przypisz 2.
3. Jeśli kwadrat liczby i jest większy od liczby n, to wypisz „liczba pierwsza” i zakończ.
4. Jeśli reszta zdzielenia n przez i wynosi zero, wypisz „liczba złożona” i zakończ.
5. Zwiększ o 1 wartość liczby pomocniczej i.
6. Przejdź do kroku 3.

Zauważcie, że zamiast badać warunek, czy wartość zmiennej i jest nie większa od liczby n, badamy, czy kwadrat zmiennej i jest nie większy od n. W ten sposób unikniemy błędów zaokrągleń, które pojawiłyby się przy zastosowaniu funkcji sqrt(n) obliczającej pierwiastek, w wyniku będącym pierwiastkiem kwadratowym liczby n. Definicja tej funkcji jest w bibliotece matematycznej cmath.

Schemat blokowy algorytmu:



Program w C+ realizujący ten algorytm:



Wyjaśnienia wymaga warunek zastosowany w pętli while, który oznacza: **wykonuj pętlę, dopóki reszta z dzielenia n przez i jest różna od zera oraz kwadrat liczby i jest mniejszy lub równy liczbie n. Wyjście z pętli nastąpi, gdy nie będzie spełniony chociaż jeden z warunków.**

**Wyznaczanie NWD i NWW dla dwóch liczb naturalnych – Algorytm Euklidesa**

Największym wspólnym dzielnikiem NWD dwóch liczb naturalnych jest największa z liczb, przez którą obie liczby dzielą się bez reszty.

Najmniejszą wspólną wielokrotnością NWW dwóch liczb naturalnych nazywamy najmniejszą z liczb, która jest podzielna przez obie te liczby.

Jak liczymy NWD?

NWD(a, b)

NWD(28, 4) //liczba a=28, b=4, sprawdzamy która jest większa, w naszym przypadku większe jest a, więc od a odejmujemy b i podstawiamy różnicę w miejsce a: 28-4=24

NWD(24, 4) //znowu a>b, więc a=a-b, a przypiszemy 24-4, czyli 20

NWD(20, 4) // a>b, więc w miejsce a podstawiamy 20-4, czyli 16

NWD(16, 4) // a>b, więc w miejsce a podstawiamy 16-4, czyli 12

NWD(12, 4) //a>b, a=12-4

NWD(8, 4) //a>b, a=8-4, to jest 4

NWD(4, 4) //koniec liczenia w momencie kiedy na miejscu a i b mamy taką samą liczbę. Obliczyliśmy, że największy wspólny dzielnik liczb 28 i 4 to 4. Taki właśnie algorytm wymyślił Euklides.

**Zadanie domowe. Sposobem który jest powyżej znajdź NWD liczb 16 i 52, sprawdź rozwiązanie w programie, zdjęcie rozwiązania pisemnego z zeszytu przyślij na** **renata.drobek@onet.eu** **do 30 maja.**

Rozwiązanie w C++



Proszę obejrzeć wykład ALGORYTMY I STRUKTURY DANYCH - Lekcja 4 - Algorytm Euklidesa - NWD i NWW dwóch liczb naturalnych. Przed wykładem zainstalować program Magiczne bloczki wersję darmową 1.0.3 , którą można pobrać ze strony

 [https://legalne.info.pl/Magiczne-Bloczki,program,download,3021,5469](https://legalne.info.pl/Magiczne-Bloczki%2Cprogram%2Cdownload%2C3021%2C5469) , ta wersja w zupełności nam wystarczy. Jeśli ktoś chce nowszą wersję to można, tylko trzeba zapłacić.

Teraz oglądamy wykład - <https://www.youtube.com/watch?v=5qbYAMOSlqI>

# oraz materiał na temat Liczb Fibonacciego - Tajemniczy ciąg Fibonacciego. Złota liczba. Boska proporcja: <https://www.youtube.com/watch?v=wb7kPaM8cfg>

Przyjemnej pracy życzę.