Algorytmy i programowanie

Marcin Żurowski

8 kwiecień 2021

Plan zajęć

1

Zapisać w postaci procedury poniższy algorytm:

```
integer a, b, k
read(a,b)
if a > b
 k = b
else
 k = a
while (a mod k) ≠ 0 or (b mod k) ≠ 0
 k = k - 1
write(k).
```

2

Zapisać w postaci procedury algorytm wyznaczający liczbę największych elementów tablicy dwuwymiarowej.

3

Podać definicję procedury wyznaczającej najmniejszą wartość tablicy `A`, przy czym chcemy, aby procedura sprawdzała dowolny podciąg od `d` do `g` kolejnych elementów tablicy, gdzie $1 \leq d \leq g \leq n$.

4

W tablicy `A` zawierającej liczby znajdują się dwa uporządkowane (niemalejąco) ciągi na miejscach od indeksu `p` do indeksu `q` i od indeksu `q + 1` do indeksu `r`, gdzie $\texttt{p} \leq \texttt{q} < \texttt{r}$. Zapisać definicję procedury, która scali te dwa ciągi w jeden niemalejący ciąg i umieści go w tablicy `A` na miejscach od `p` do `r`.

5

W tablicy `A` zawierającej liczby znajdują się dwa uporządkowane (niemalejąco) ciągi na miejscach odpowiednio od indeksu `p` do indeksu `q` i od indeksu `q + 1` do indeksu `r`, gdzie $\texttt{p} \leq \texttt{q} < \texttt{r}$. Zapisać definicję procedury, która scali te dwa ciągi w jeden niemalejący ciąg i umieści go w tablicy `A` na miejscach od `A[p]` do `A[r]` bez sprawdzania za każdym razem, czy wszystkie elementy któregoś z ciągów zostały już wzięte pod uwagę.

6

Zadana jest tablica `A[p..r]` zawierająca liczby. Napisać definicję procedury, która dzieli tę tablicę (poprzez przestawienie jej elementów) na dwie tablice `A[p..q - 1]` i `A[q + 1..r]` w ten sposób, że każdy element z pierwszej podtablicy jest nie większy niż element `A[q]`, który z kolei jest mniejszy od każdego elementu z drugiej podtablicy. Obliczenie indeksu `q` ma stanowić część tej procedury podziału.

7

Zapisać definicję funkcji wyznaczającej najmniejszy element tablicy jednowymiarowej.

8

Zapisać definicję procedury `NWD` w postaci procedury funkcyjnej.

9

Dane są dwa wektory `A[1..n]` `B[1..n]` zawierające liczby.
Napisać definicję funkcji logicznej przyjmującej wartość `true` wtedy i tylko wtedy, gdy oba wektory są równe.

Rekurencja

```
#include <iostream>
using namespace std;
int fib(int n);
int main() {
    cout << fib(9) << endl;
    return 0;
}
```

Rekurencja

```
int fib(int n) {
    // warunek stopu
    if (n == 1 || n == 2) {
        return 1;
    }
    // obliczanie wyniku pośrednich
    int wPop = fib(n - 1);
    int wPopPop = fib(n - 2);
    // łączenie wyników w wynik końcowy
    return wPop + wPopPop;
}
```