Politechnika Wrocławska

Wyszukiwanie ze wstawianiem

Wojciech Myszka

Katedra Mechaniki, Inżynierii Materiałowej i Biomedycznej

2022-04-03 10:22:50 +0200

HR EXCELLENCE IN RESEARCH

Ogólny schemat

1. Zaczynamy z **pustą** tablicą ($N = 0$).
2. Gdy przychodzi pierwsza liczba zwiększamy tablicę do rozmiaru 1 i dodajemy liczbę.
3. Gdy przychodzi kolejna liczba, sprawdzamy czy jest już w tablicy
   - jeżeli TAK przechodzimy do punktu 3
   - jeżeli NIE
     - 3.1 zwiększamy rozmiar tablicy o 1
     - 3.2 dodajemy element
     - 3.3 zwiększamy N o 1 ($N = N + 1$)
     - 3.4 drukujemy tablicę

   przechodzimy do punktu 3

Do rozstrzygnięcia pozostaje jak przerwać program. Proponuję, żeby założyć, że wszystkie wartości są dodatnie, jak przyjdzie wartość ujemna — zatrzymujemy pracę programu.

Odczyt liczby

Do wprowadzania użyjemy funkcji scanf():

```
int x;
// ...
scanf("%d", &x);
if (x < 0)
{
    printf("Koniec programu!");
    return 0;
}
```

Wyszukiwanie

1. W większości przypadków Państwa programy były OK.
2. Zadbać trzeba o to, żeby funkcja zwracała
   - wartość dodatnią lub zero gdy liczba została znaleziona (oznaczająca miejsce, na którym liczba się znajduje)
   - wartość ujemną określającą pozycję, na której liczba ma być dodana, według kodu:
     -1 — miejsce zerowe
     -2 — miejsce pierwsze
     ...
     $-n$ — wartość ma być wstawiona na miejscu $n-1$

Uwaga: Można oczywiście użyć innego sposobu kodowania!

Politechnika Wrocławska

„Zaczynamy z pustą tablicą”

```
int N = 0;
int * dane;
```

1. Korzystamy ze wskaźników (tylko to pozwala na zwiększanie długości tablicy)

Politechnika Wrocławska

„Zwiększamy tablicę do rozmiaru 1...”

```
dane = malloc((N + 1) * sizeof(int));
if (dane == 0)
{
    printf("Blad! Nie dostalem pamieci!\n");
    return 1;
}
N = N + 1;
```

Politechnika Wrocławska

„Zwiększamy rozmiar tablicy o 1”

```
int * dane1;
dane1 = realloc(dane, (N + 1) * sizeof(int));
if (dane1 == 0)
{
    printf("Blad! Nie dostalem pamieci!\n");
    return 1;
}
dane = dane1;
```

Użycie nowej zmiennej (dane1) ma sens, gdyż, w sytuacji gdy nie dostaniemy pamięci — funkcja realloc() zwraca wartość zero. Mamy szanse coś z dotychczas zebranymi danymi zrobić. Tu program kończy pracę.

Dodawanie liczby I

- Zakładam, że tablica ma wystarczającą długość (została „przedłużona” przed wywołaniem funkcji)
- Wiemy ile jest liczb w (starej) tablicy (N)
- Mamy informację na którym miejscu liczbę dodać (k)
  - gdy $k < N$ — „gdzieś między liczby w tablicy”
  - gdy $k == N$ — jako ostatnią wartość

Podczas wstawiania trzeba będzie (czasami) liczby w tablicy przesuwać!
Prototyp funkcji

```
void wstaw(int x, int k, int N, int * T);
// x wstawiana liczba; k miejsce wstawiania
// N dlugosc tablicy;  T tablica danych
```

Dodawanie liczby II

Algorytm:

1. Gdy $k == N$ liczbę trzeba wstawić na końcu (nie trzeba przesuwać danych)
   - `T[k] = x;`
   - koniec
2. W przeciwnym razie trzeba przesunąć wszystkie wartości o indeksach większych niż k „w prawo” (najlepiej robić to od końca)

```
for(int i = N; i > k; i = i - 1)
    T[i] = T[i - 1];
T[k] = x;
```

3. Koniec

Drukowanie tablicy

Łatwizna.
Prototyp funkcji:

```
void drukuj(int N, int * T);
```