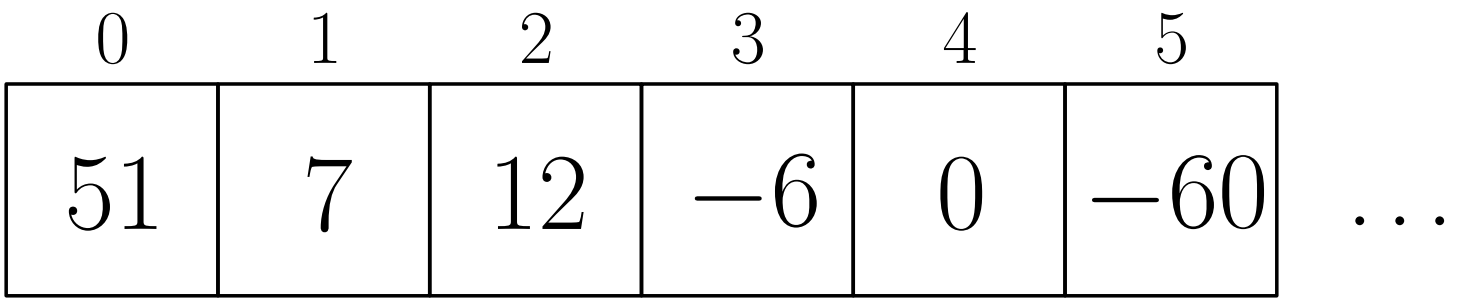
Procvičování algoritmizace s polem

David Weber

15. září 2022

Opakování

- datová struktura, která sdružuje dané prvky (čísla, textové řetězce, . . .) **stejného datového typu**.
- Velikost pole zůstává při běhu programu **neměnná**.
- K jednotlivým prvkům přistupujeme pomocí tzv. *indexu* (celé číslo od 0 do $n - 1$.)



Na rozehřátí

Na rozehřátí

- **Úloha:** Napište algoritmus, který pro libovolné pole obsahující prvky typu *integer* nalezne **minimum** a **maximum** a vypíše jej.

Na rozehřátí

- **Úloha:** Napište algoritmus, který pro libovolné pole obsahující prvky typu *integer* nalezne **minimum** a **maximum** a vypíše jej.
- **Úloha:** Napište algoritmus, který pro **uživatelem zadané číslo** vyhledá a vypíše index prvního jeho výskytu v poli. Pokud se zadaný prvek v poli nenachází, vypíše -1.

Vzorové řešení první úlohy

```
int min = A[0]; int max = A[0];
int n = sizeof(A)/sizeof(int);

for (int i=0; i<n; i++){
    if (A[i] < min) min = A[i];
    if (A[i] > max) max = A[i];
}
```

Vzorové řešení druhé úlohy

```
int element;
scanf("%d", &element);
int n = sizeof(A)/sizeof(int);
int index = -1;

for (int i=0; i<n; i++){
    if (A[i] == element){
        printf("%d\n", i);
        break;
    }
}
```

Nyní těžší úlohy...

Nyní těžší úlohy...

- **Úloha:** Napište algoritmus, který pro uživatelem zadané číslo S vypíše **všechny** dvojice indexů i, j v poli A, takové, že

$$A[i] + A[j] = S.$$

(Tj. příslušná dvojice prvků je rovna zadanému součtu.)

Nyní těžší úlohy...

- **Úloha:** Napište algoritmus, který pro uživatelem zadané číslo S vypíše **všechny** dvojice indexů i, j v poli A, takové, že

$$A[i] + A[j] = S.$$

  (Tj. příslušná dvojice prvků je rovna zadanému součtu.)
- Zaveďte do svého algoritmu počítadlo kroků, tak, aby se zvýšilo při každém provedeném porovnání a na konci jej vypište.

```
int sum;
scanf("%d", &sum);
int n = sizeof(A)/sizeof(int);

for (int i=0; i<n; i++){
    for (int j=i+1; j<n; j++){
        if (A[i] + A[j]==sum)
            printf("i=%d, j=%d\n", i, j);
    }
}
```

Optimalizace

Optimalizace

- Při návrhu algoritmu se vždy snažíme ideálně o co nejoptimálnější řešení.

Optimalizace

- Při návrhu algoritmu se vždy snažíme ideálně o co nejoptimálnější řešení.
- Naším cílem tak je dosáhnout správného výsledku pomocí co nejméně kroků $\Longrightarrow$ šetříme čas a algoritmus lze použít i na větší vstupy.

Optimalizace

- Při návrhu algoritmu se vždy snažíme ideálně o co nejoptimálnější řešení.
- Naším cílem tak je dosáhnout správného výsledku pomocí co nejméně kroků $\implies$ šetříme čas a algoritmus lze použít i na větší vstupy.

Byl by Vámi navržený algoritmus optimální, pokud bychom pracovali se **setříděným polem**? Můžeme této vlastnosti nějak využít?