Programowanie C++

Wykład 3 (16.03.2016)

Uwaga: slajdy w znacznej części bazują na wykładzie Rafała Wysockiego

Tablice, elementy, indeksy

Tablica (*ang. array*)

Zestaw N zmiennych tego samego typu numerowanych liczbami w zakresie od 0 do $(N-1)$.

Element tablicy

Zmienna wchodząca w skład tablicy, mająca przypisaną określoną liczbę, która ją identyfikuje.

Indeks (*ang. index*) elmentu tablicy

Liczba identyfikująca element tablicy.

Długość tablicy

Liczba elementów tablicy (N).

Deklaracja i definicja tablicy

Deklaracja tablicy

Określenie typu danych dla elementów tablicy, nazwy oraz (nie obowiązkowo) liczby elementów w nawiasie kwadratowym, np.

```
int tab[100];
int liczby[];
```

Definicja tablicy

Deklaracja tablicy, w której jest określona liczba elementów.

Po zdefiniowaniu tablicy każdy jej element może być wykorzystywany jako niezależna zmienna.

Tablica – przykład

```
int tab[10];
...
for (int j = 0; j < 10; j++)
    tab[j] = 0;
```

Uwaga!

1. Kompilator C++ nie sprawdza, czy używane w programach wartości indeksów tablic są właściwe.
2. Wartości te również nie są kontrolowane podczas wykonywania programu.
3. Ujemne wartości indeksów są dopuszczalne (oznaczają hipotetyczne elementy o adresach mniejszych od adresu pierwszego elementu).

Przykład zastosowania tablicy

Tablice dobrze sprawdzają się w zastosowaniach, w których mamy do czynienia ze stosunkowo dużą, ustaloną liczbą obiektów tego samego typu.

Problem

W zbiorze $(N + M + 1)$ wylosowanych liczb mamy znaleźć taką, dla której w tym zbiorze jest N liczb nie większych i M nie mniejszych od niej. Dla $N = M$ jest to problem wyznaczania mediany („środkowego elementu") zbioru.

Metoda poszukiwania rozwiązania

Ustawimy liczby w danym zbiorze w kolejności rosnącej i wybierzemy tę, która będzie na pozycji $(N + 1)$.

Algorytm sortowania tablicy (przez wybieranie)

1. Zapisz każdą wylosowaną liczbę w innym elemencie tablicy $a[]$ o długości $L = N + M + 1$.
2. Powtarzaj dla indeksów i od 0 do $(L - 2)$:
   1. Wyznacz indeks j taki, że element tablicy o tym indeksie jest najmniejszy spośród elementów o indeksach od i do $(L - 1)$ włącznie.
   2. Zamień miejscami element o indeksie i z elementem o indeksie j.
3. Po zakończeniu powyższych czynności tablica jest posortowana.

Wyznaczanie najmniejszego elementu (krok 2.1)

1. Niech $j = i$.
2. Powtarzaj dla indeksów k od $(i + 1)$ do $(L - 1)$:
   - Jeśli $a[k] < a[j]$, to niech $j = k$.
3. j jest poszukiwanym indeksem.

Program sortujący tablicę

```
for (int i = 0; i < L-1; i++) {
    // Poszukiwanie indeksu j
    int j = i;
    for (int k = i + 1; k < L; k++)
        if (a[k] < a[j])
            j = k;

    // Zamiana miejscami a[i] z a[j]
    if (j > i) {
        double m = a[j];
        a[j] = a[i];
        a[i] = m;
    }
}
```

Program sortujący tablicę – c. d.

Wypełnianie tablicy

W powyższym kodzie źródłowym pominięto wypełnianie tablicy:

```
srandom(time(NULL));
for (int i = 0; i < L; i++)
    a[i] = 1.0 / (1.0 + random());
```

Funkcja `random()` generuje liczby pseudolosowe w zakresie od 0 do `RAND_MAX`. Do tablicy wstawiane są liczby pseudolosowe z przedziału $(0, 1]$.

Wynik obliczeń

Po przeprowadzeniu sortowania tablicy wystarczy wydrukować $a[N]$ jako wynik.

Sito Eratostenesa

Poszukujemy liczb pierwszych nie większych od N

Niech $A = \{2, 3, ..., N\}$ będzie zbiorem liczb, natomiast k i m – miejscami do przechowywania pośrednich wyników.

1. Niech $k = 2$.
2. Jeśli $k^2 > N$, zbiór A zawiera tylko liczby pierwsze.
3. Usuwamy z A wszystkie wielokrotności k, począwszy od k^2.
4. W m zapisz najmniejszą liczbą ze zbioru A większą od k.
5. Niech $k = m$.
6. Przejdź do kroku 2.

Zbiór A w powyższym algorytmie można zastąpić tablicą.

Sito Eratostenesa z użyciem tablicy

Poszukujemy liczb pierwszych nie większych od N

Niech $a[]$ będzie tablicą o $(N+1)$ elementach typu `bool`.

1. Niech wszystkie elementy $a[j]$ dla $j > 1$ mają wartość `true`.
2. Niech $k = 2$
3. Wstaw `false` do wszystkich elementów $a[j]$, dla których j jest wielokrotnością k, począwszy od k^2.
4. Powtarzaj:
   - $k = k + 1$
   - Jeśli $k^2 > N$, przejdź do kroku 5.
   - Jeśli $a[k]$ ma wartość `true`, przejdź do kroku 3.
5. Wydrukuj indeksy j, dla których elementy $a[j]$ mają wartość `true`.

Sito Eratostenesa z użyciem tablicy – program

```
bool a[N+1];
int j;

for (j = 2; j <= N; j++)
    a[j] = true;

for (int k = 2; k*k <= N; k++)
    if (a[k]) {
        for (j = k*k; j <= N; j += k)
            a[j] = false;
    }

for (j = 2; j <= N; j++)
    if (a[j])
        cout << j << endl;
```

Czym są wskaźniki

Wskaźnik (*ang. pointer*)

Zmienna, której wartością jest adres innej zmiennej.

Deklaracja wskaźnika

Umieszcza się * przy nazwie zmiennej, np.:

```
int *ptr;  // Wartością jest adres zmiennej typu int
char *p;  // Wartością jest adres zmiennej typu char
```

Operator & – obliczanie adresu zmiennej

`n` – zmienna typu `int`, `ptr` – wskaźnik zdefiniowany jak wyżej.

```
ptr = &n;  // Adres n staje się wartością wskaźnika ptr
```

Dostęp do zmiennej poprzez wskaźnik

Gdy wskaźnik `wsk` zawiera adres zmiennej `n`

Wtedy następujące operacje dają ten sam wynik (wstawienie liczby 2 do zmiennej `n`):

- `n = 2;`
- `*wsk = 2;`

Operację oznaczoną symbolem * powyżej nazywa się wyłuskaniem.

Wskaźnik można traktować jak „okno" dające dostęp do zmiennej.

Dostęp do elementów tablicy poprzez wskaźnik

Elementy tablicy i wskaźnik tego samego typu

```
ptr = &a[j];  // Adres a[j] staje się wartością wskaźnika ptr
ptr + 1  // Adres a[j+1]
ptr - 1  // Adres a[j-1]
ptr + k  // Adres a[j+k]
```

Dla ptr == &a[j] równoważne są następujące zapisy

```
a[j+k]   // Wartość elementu a[] o indeksie j+k
*(ptr+k)   // Wartość zmiennej pod adresem ptr przesuniętym o k
ptr[k]   // Wartość zmiennej pod adresem ptr przesuniętym o k
*(a+j+k)   // Wartość zmiennej pod adresem a przesuniętym o j+k
```

Nazwę tablicy można traktować jako stałą wskaźnikową.

Operacje na wskaźnikach

Zwiększanie i zmniejszanie, += i -=

- Prawa strona musi być całkowitego typu.
- Oznacza przesunięcie adresu we wskaźniku o pewną liczbę pozycji.
- Dla wskaźnika typu `int`

```
ptr += k;  // Adres w ptr jest zwiększany o k * sizeof(int)
```

Inkrementacja i dekrementacja, ++ i --

- Przesunięcie „do przodu" lub „do tyłu" o jedną zmienną danego typu (np. element tablicy).
- Można łączyć z innymi wyrażeniami (jak dla „zwykłych" zmiennych).
- Dla wskaźnika typu `int`

```
ptr++;  // Adres w ptr jest zwiększany o sizeof(int)
```

Odejmowanie wskaźników

- Można odjąć jeden wskaźnik od drugiego (tego samego typu).
- Ma to sens, gdy wskaźniki zawierają adresy elementów tej samej tablicy.
- Wynikiem jest różnica między adresami wyrażona jako przesunięcie mierzone liczbą pozycji w tablicy (tzn. różnica indeksów odpowiadających elementom tablicy, których adresy zawierają te wskaźniki).

Rezerwowanie pamięci na żądanie

Rezerwowanie pamięci z pomocą `new`

- Rezerwowanie pamięci na zmienną:

```
ptr = new double;
```

- Rezerwowanie pamięci na tablicę:

```
ptr = new double[N];
```

Zasady

1. Typ danych wskaźnika `ptr` musi odpowiadać typowi danych rezerwowanej zmiennej lub tablicy (`void *` „pasuje" do każdego typu danych).
2. W przypadku niepowodzenia wskaźnik `ptr` będzie mieć wartość `NULL` (czyli 0).

Zwalnianie pamięci przydzielonej na żądanie

Pamięć przydzielona na żądanie pozostaje do dyspozycji programu, aż zostanie przez niego jawnie zwolniona (nie ma znaczenia to, w którym miejscu programu została ona zarezerwowana).

Zwalnianie pamięci z pomocą `delete`

- Zwalnianie pamięci zarezerwowanej na zmienną:

```
delete ptr;
```

- Zwalnianie pamięci zarezerwowanej na tablicę:

```
delete [] ptr;
```

Wskaźnik `ptr` musi zawierać adres obszaru pamięci przydzielonego na żądanie.

Problem z przekazywaniem tablic do funkcji

W C++ tablica nie może być argumentem funkcji

1. Argumenty funkcji to zmienne:
   - Których wartości początkowe pochodzą z innych części programu (przekazywanie przez wartość).
   - Które zostały zdefiniowane w innych częściach programu (przekazywanie przez referencję).
2. Tablice nie są zmiennymi!
   - Tablica jest zespołem zmiennych.
   - Nazwa tablicy jest stałą wskaźnikową.

Dostęp do tablic (z funkcji) poprzez wskaźniki

Wskaźniki mogą być argumentami funkcji

1. Wskaźnikowe argumenty funkcji można w treści tej funkcji traktować tak, jakby były nazwami tablic.
2. Wartość (początkowa) parametru wskaźnikowego oznacza lokację (w pamięci), którą funkcja ma traktować jako początek tablicy.
3. Funkcja musi „poznać" liczbę elementów tablicy niezależnie od adresu jej początku.
   - Informacja o liczbie elementów nie jest automatycznie przekazywana wraz z adresem początku tablicy.
   - Musi ona być przekazana do funkcji w inny sposób (np. jako dodatkowy argument funkcji).

```
void sort(double *a, int n);
```