

Тема 7.

Програмування структурованих типів даних в C++

Заняття 1. Масиви

Література до заняття



C++. Основи програмування. Теорія та практика : підручник / [О. Г. Трофименко, Ю.В. Прокоп, І.Г. Швайко, Л.М. Буката та ін.] ; за ред. О.Г. Трофименко. – Одеса: Фенікс, 2011. – 587 с.

Література

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Ю. А. Белов
Т. О. Карнаух
Ю. В. Коваль
А. Б. Ставровський

ВСТУП ДО ПРОГРАМУВАННЯ МОВОЮ C++

ОРГАНІЗАЦІЯ ОБЧИСЛЕНЬ

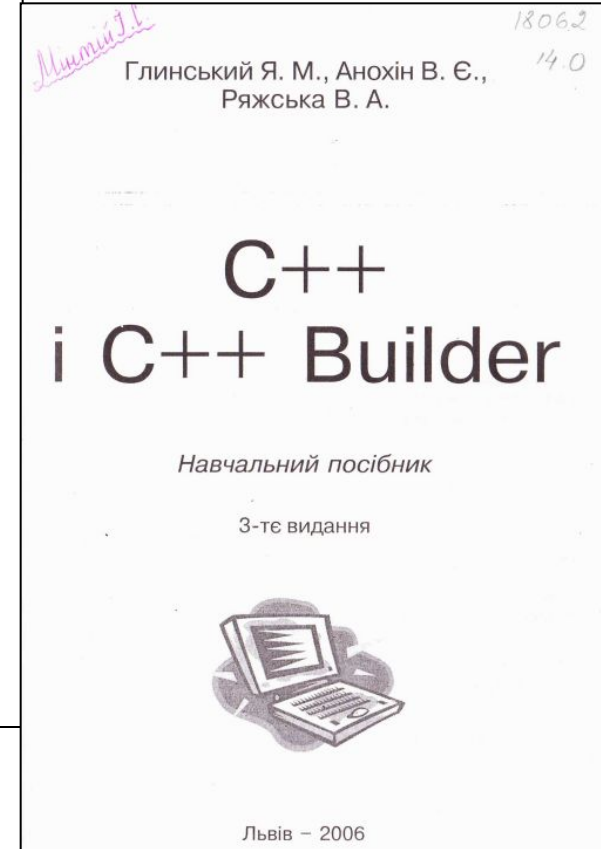
Навчальний посібник

Міністерство надзвичайних ситуацій України
Львівський державний університет безпеки життєдіяльності

Юрій ГРИЦЮК,
Тарас РАК

ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ МОВОЮ C++

Навчальний посібник



КИЇВСЬКИЙ

В.В.Войтенко, А.В.Морозов

C/C++ теорія та практика

Частина 2

Мова програмування C++

2003

Львів
Вид-во ЛДУ БЖД
2011

Н. Культин

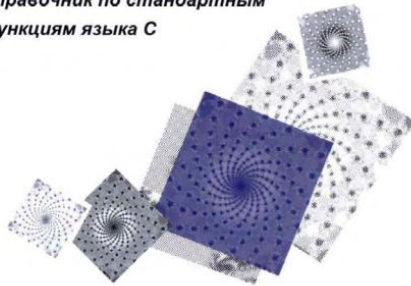


C/C++ в задачах и примерах

Примеры
и исходные тексты
программ с комментариями

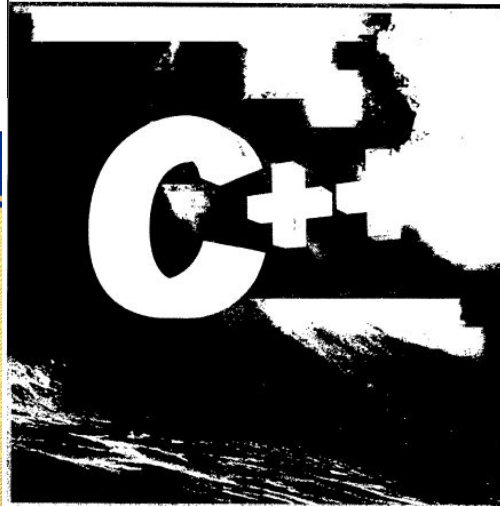
Более
250 задач

Справочник по стандартным
функциям языка C



Літератур

ЯЗЫК C++ ПРОГРАММИРОВАНИЯ специальное издание



Бьери Страуструп
создатель C++

Никита Культин



C++ Builder в задачах и примерах

Использование
базовых компонентов

Программирование
графики, игр, мультимедиа
и баз данных

Справочник
по компонентам
и функциям

Готовые решения
и тексты программ







УЧЕБНОЕ ПОСОБИЕ

Т. А. Павловская, Ю. А. Щупак

C++

Объектно-ориентированное
программирование


Практикум



НАУКА И КОМПЬЮТЕРНАЯ НАУКА

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ В C++

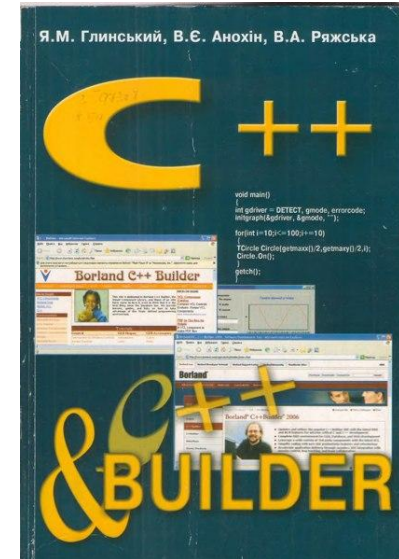
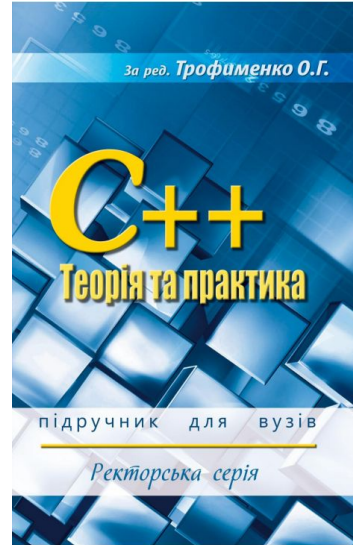
4-Е ИЗДАНИЕ



Р. ЛАФОРЕ

Література до заняття



- Вступ до програмування мовою C++. Організація обчислень: навч. посіб. / Ю. А. Белов, Т. О. Карнаух, Ю. В. Коваль, А. Б. Ставровський. – К. : Видавничо-поліграфічний центр "Київський університет", 2012. – 175 с.
- C++. Основи програмування. Теорія та практика : підручник / [О.Г. Трофименко, Ю.В. Прокоп, І.Г. Швайко, Л.М. Буката та ін.] ; за ред. О.Г. Трофименко. – Одеса: Фенікс, 2010. – 544 с.
- Глинський Я.М., Анохін В.С., Ряжська В.А. C++ і C++ Builder. Навч. посібник. 3-тє вид. –Львів: СПД Глинський, 2006. – 192 с.

Масиви

Масиви – це група однотипних елементів які мають загальне ім'я і розміщені в пам'яті один біля одного.

Особливості:

- всі елементи мають **один тип**
- весь масив має **одне ім'я**
- всі елементи розміщені в пам'яті **один біля одного**

Приклади:

- Список курсантів в групі
- Квартири в будинку
- Школи в місті
- Дані про температуру повітря за рік

Масиви

A

масив

НОМЕР

Елемента масива
(ІНДЕКС)

0

1

2

3

4

5

10

15

20

25

A[0]

A[1]

ЗНАЧЕННЯ

Елемента масива

A[4]

ЗНАЧЕННЯ

Елемента масива: 15

A[2]

НОМЕР(ІНДЕКС)

елемента масива: 2



Нумерація елементів масиві в Сі з **НУЛЯ!**

Оголошення масивів

Навіщо оголошувати?

- Визначити **ім'я** масива
- Визначити **тип** масива
- Визначити **число елементів**
- Виділити **місце в пам'яті**

Приклад:

Тип
елементів

Ім'я

Розмір масива
(кількість
елементів)

```
int A [ 5 ] ;
```

Розмір через константу:

```
const int N =  
5;  
int A [ N ] ;
```


Оголошення масивів

Ще приклади:

```
int X[10], Y[10];  
float zz, A[20];  
char s[80];
```

З присвоюванням початкових значень :

```
int A[4] = { 8, -3, 4, 6 };  
float B[2] = { 1. };  
char C[3] = { 'A', '1', 'Ю' };
```

Решта
нульові!



Якщо початкові значення не задані то в чарунках знаходиться «СМІТТЯ»!

Що неправильно?

```
const int N = 10;
float A[N];
```

```
int X[4.5];
```

```
int A[10];
A[10] = 0;
```

Виходить за
рамки масива
(стираються дані в
пам'яті)

```
float X[5];
int n = 1;
X[n-2] = 4.5;
X[n+8] = 12.;
```

Дробова частина
відкидається
(помилки немає)

```
int X[4];
X[2] = 4.5;
```

```
float A[2] = { 1, 3.8 };
```

```
float B[2] = { 1., 3.8, 5.5 };
```

Масиви

Оголошення :

```
const int N = 5;
int A[N], i;
```

Введення з клавіатури:

```
cout << "Введіть 5 елементів" << endl;
for( i=0; i<N; i++ ) {
    cout << " A[" << i << "]=";
    cin >> a[i] >> endl;
}
```

A[0] = 5
 A[1] = 12
 A[2] = 34
 A[3] = 56
 A[4] = 13

По

Ви

```
for( i=0; i<N; i++ ) A[i] = A[i]*2;
```

```
cout << "Результат:" << endl;
for( i=0; i<N; i++ )
    cout << A[i] << " ";
cout << endl;
```

Результат:

10 24 68 112 26



Як вивести в стовпчик?

Програма

Завдання: ввести з клавіатури масив з 5 елементів, помножити всі елементи на 2 і вивести отриманий масив на екран.

```
#include <iostream.h>
#include <conio.h>
main()
{
  const int N = 5;
  int A[N], i;
  // введення елементів масива
  // обробка масива
  // виведення масива
  getch();
}
```

На попередніх
слайдах

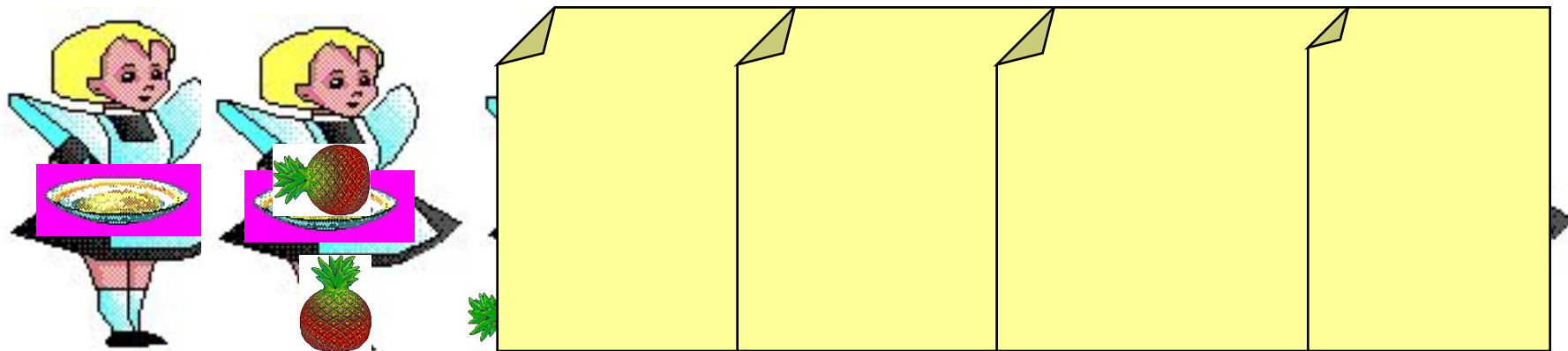
Програмування мовою C++

**Тема . Максимальний
елемент масива**

Максимальний елемент

Завдання: знайти в масиві максимальний елемент.

Алгоритм:



Псевдокод :

```
// рахуємо, що елемент A[0] – максимальний  
for ( i=1; i < N; i++ )  
    if ( A[i] > максимального )  
        // запам'ятати новий максимальний елемент A[i]
```



Чому цикл від $i=1$?

Максимальний елемент

Додаток : як знайти номер максимального елемента?

```
        // поки A[0] – максимальний
iMax = 0;
for ( i=1; i < N; i++ ) // перевіряємо інші
    if ( A[i] > A[iMax] ) { // знайшли новий
        // запам'ятати A[i]
        iMax = i; // запам'ятати i
    }
```



Як спростити?

По номеру елемента **iMax** завжди можна знайти його значення **A[iMax]**. Тому всюди міняємо **max** на **A[iMax]** і прибираємо змінну **max**.

Заповнення випадковими числами

```
#include <stdlib.h> // випадкові числа
```

RAND_MAX – максимальне випадкове число
(зазвичай RAND_MAX = 32767)

Ініціалізація генератору випадкових чисел

```
randomize (); // запуск генератору
```

Випадкове ціле число в інтервалі [0,RAND_MAX]

```
x = random (); //перше число
```

```
x = random (); // вже інше число
```

Встановити початкове значення послідовності :

```
random ( 345 ); // починаємо з 0
```


Цілі числа в заданому інтервалі

Цілі числа в інтервалі $[0, N-1]$:

```
int random (int N) {  
    return rand() % N;  
}
```

Приклади :

```
x = random ( 100 ) ; // інтервал [0, 99]  
x = random ( z ) ; // інтервал [0, z-1]  
x = (50-random(100)) ; // інтервал [-49, 50]  
int x = RandomRange (15, 20) ; // інтервал [15, 20]
```

Цілі числа в інтервалі $[a, b]$:

```
x = random ( z ) + a ; // інтервал [a, z-1+a]  
x = random ( b - a + 1 ) + a ; // інтервал [a, b]
```

Заповнення випадковими числами

```
#include <iostream.h>
#include <stdlib.h>

int random(int N)
{ return rand() % N; }

main()
{randomize();
const int N = 10;
int A[N], i;
cout << "Початковий масив:\n";
for (i = 0; i < N; i++) {
    A[i] = random(100) + 50;
    cout << "A[i]=" << A[i]\n;
}
...
}
```

Функція видає
випадкове число
від 0 до N-1



Який інтервал?

Програма

```
#include <iostream.h>
```

```
#include <stdlib.h>
```

```
main()
```



Що дає `const`?

```
{
```

```
const int N = 5;
```

```
int A[N], i, iMax;
```

```
    // запам'ятати випадкові числа [100,150]
```

```
    // знайти максимальний елемент і його номер
```

```
cout << "\t Максимальний елемент A[" << iMax;
```

```
cout << "]" << A[iMax] \n;
```

```
getch();
```

```
}
```

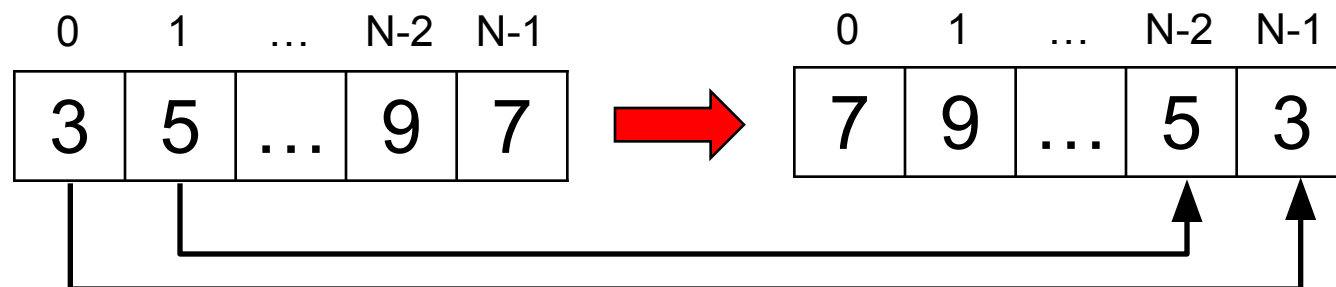
На попередніх
слайдах

Програмування мовою C++

Тема . Обробка масивів

Реверс масива

Завдання: переставити елементи масива в обернутому порядку (виконати інверсію).



Алгоритм:

Поміняти місцями $A[0]$ и $A[N-1]$, $A[1]$ и $A[N-2]$, ...

Псевдокод :

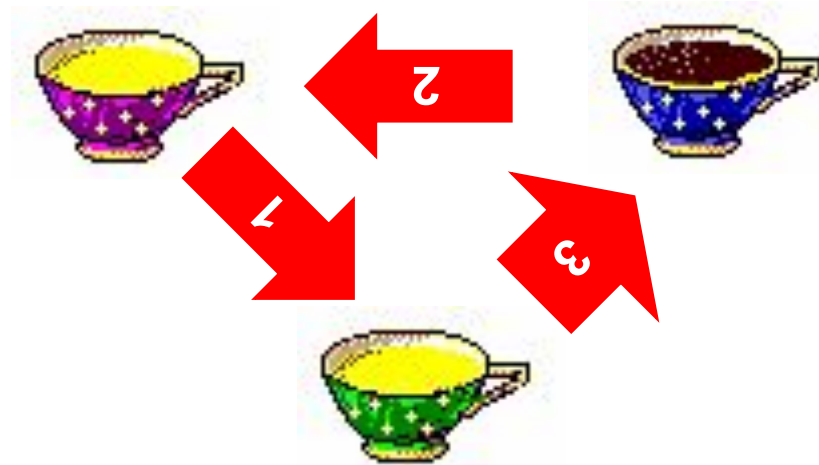
```
for ( i = 0; i < N / 2 ; i++ )
  // помінти місцями A[i] и A[N-1-i]
```



Що неправильно?

Як переставити елементи?

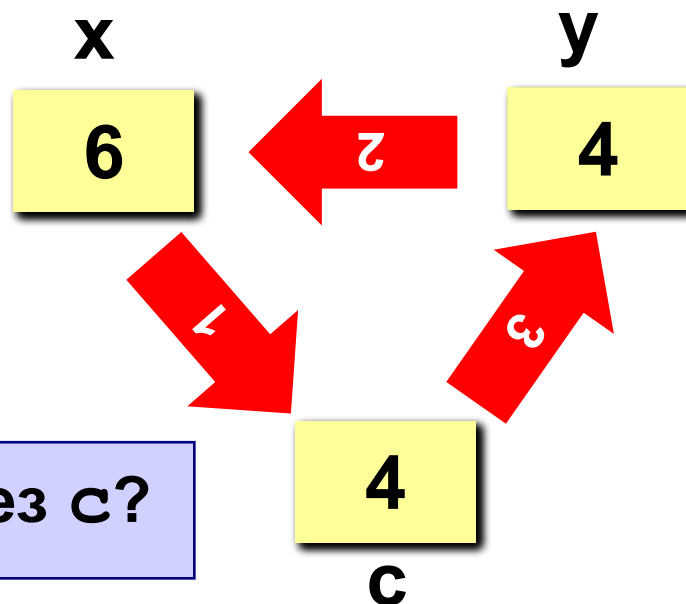
Завдання: поміняти місцями вміст двох чашок .



Завдання: поміняти місцями вміст двох чарунок пам'яті.

~~x = y;
y = x;~~

c = x;
x = y;
y = c;



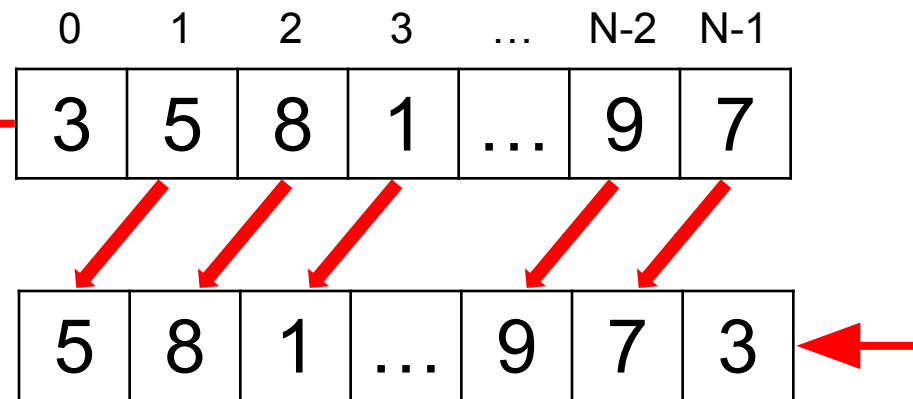
Чи можливо обійтися без c?

Програма

```
main ()
{
    const int N=10;
    int A[N], i, c;
    // заповнити масив
    // вивести початковий масив
    for ( i=0; i<N/2; i++ ) {
        c=A[i];
        A[i]=A[N-1-i];
        A[N-1-i]=c;
    }
    // вивести отриманий масив
}
```

Циклічний зсув

Завдання : зсунути елементи масива вліво на 1 чарунку, перший елемент стає на місце остатнього .



Алгоритм:

$A[0]=A[1] ; A[1]=A[2] ; \dots A[N-2]=A[N-1] ;$

Цикл:

Чому не N?

```
for ( i = 0; i < N-1; i++)
    A[i] = A[i+1];
```



Що неправильно?

Програма

```
main()  
{  
    const int N = 10;  
    int A[N], i, c;  
    // заповнити масив  
    // вивести початковий масив  
    c = A[0];  
    for ( i = 0; i < N-1; i ++ )  
        A[i] = A[i+1];  
    A[N-1] = c;  
    // вивести новий масив  
}
```

Програмування мовою C++

Тема. Сортування масивів

Сортування

Сортування це розстановка елементів масива в заданому порядку (по збільшенню, зменшенню..)

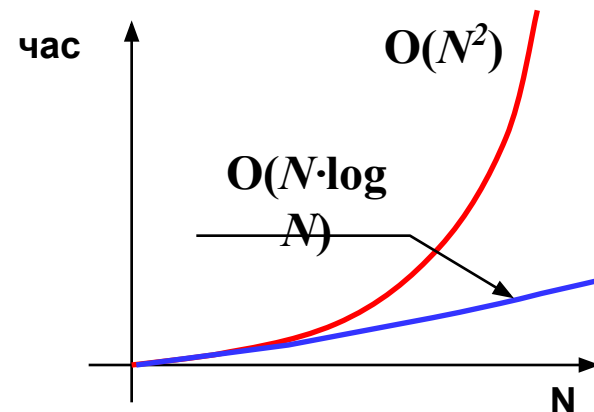
Задача : переставити елементи масива в порядку збільшення .

Алгоритми :

складність $O(N^2)$

- Прості й зрозумілі але не ефективні для великих масивів
 - **Метод бульбашки**
 - **Метод вибора**
- Складні але ефективні
 - «швидке сортування» (*Quick Sort*)
 - сортування «кучею» (*Heap Sort*)
 - сортування злиттям
 - пірамідальне сортування

складність $O(N \cdot \log N)$

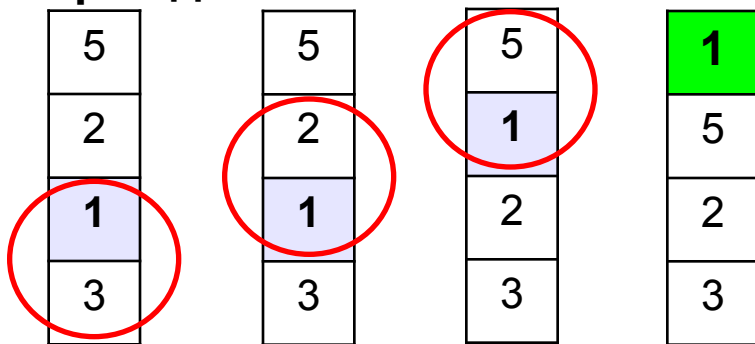


Метод бульбашки

Ідея – бульбашка в стакані з водою з дна піднімається вверху .

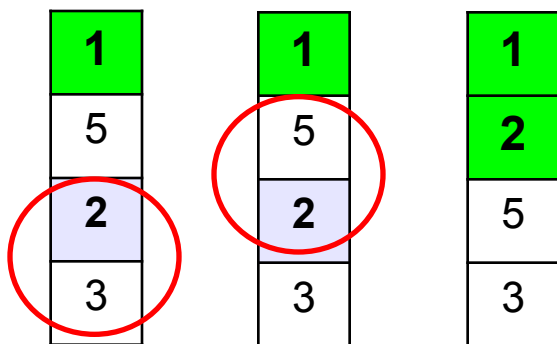
Для масивів – самий маленький («легкий») елемент переміщується вгору («вспливає»).

1-ший прохід

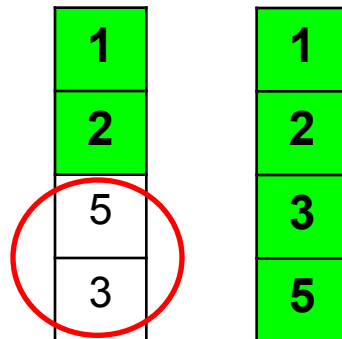


- Починається знизу, порівнюємо два сусідніх елемента; якщо вони стоять «неправильно», міняємо їх місцями
- Через 1 прохід по масиву **один** елемент (самий маленький) стає на своє місце

2-й прохід



3-й прохід



Для сортування масива із N елементів потрібен $N-1$ прохід (достатньо поставити на своє місце $N-1$ елементів).

Програма (1-ий прохід)

0	5
1	2
...	...
N-2	6
N-1	3

Порівнюються пари

$A[N-2]$ та $A[N-1]$,

$A[N-3]$ та $A[N-2]$

...

$A[0]$ та $A[1]$

$A[j]$ та $A[j+1]$

```
for ( j = N-2; j >= 0; j-- )
    if ( A[j] > A[j+1] ) {
        c = A[j];
        A[j] = A[j+1];
        A[j+1] = c;
    }
```

Програма (наступні проходи)

2-ий
прохід

0	1
1	5
...	...
N-2	3
N-1	6

(i+1)-ий
прохід



A[0] вже на своєму місці!

```
for ( j = N-2; j >= 1; j-- )
    if ( A[j] > A[j+1] ) {
        c = A[j];
        A[j] = A[j+1];
        A[j+1] = c;
    }
```

```
for ( j = N-2; j >= i; j-- )
    ...
```

Програма

```

main ()
{
  const int N = 10;
  int A[N], i, j, c;
  // заповнити масив
  // вивести початковий масив
  for (i = 0; i < N-1; i++) {
    for (j = N-2; j >= i; j--)
      if (A[j] > A[j+1]) {
        c = A[j];
        A[j] = A[j+1];
        A[j+1] = c;
      }
  }
  // вивести новий масив
}

```



чому для циклу $i < N-1$,
а не $i < N$?

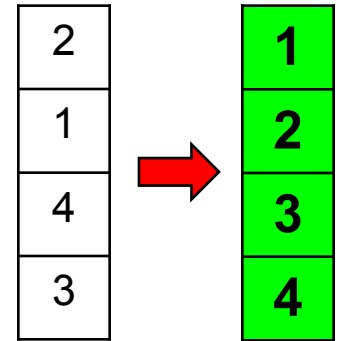
Елементи вище
 $A[i]$ вже
поставлені

Міняємо
 $A[j]$ та
 $A[j+1]$

Метод бульбашки з прапорцем

Ідея – якщо при виконанні метода бульбашки не було обмінів, масив вже відсортований і решта проходів непотрібні.

Реалізація: змінна-прапорець, вказуюча, чи був обмін; якщо вона дорівнює 0, тоді вихід.



```

do {
    int flag;
    flag = 0; // скинути прапорець
    for (j = N-2; j >= 0; j --)
        if (A[j] > A[j+1]) {
            c = A[j];
            A[j] = A[j+1];
            A[j+1] = c;
            flag = 1; // підняти прапорець
        }
    }
while ( flag ); // вихід якщо flag = 0

```

? Як покращити?

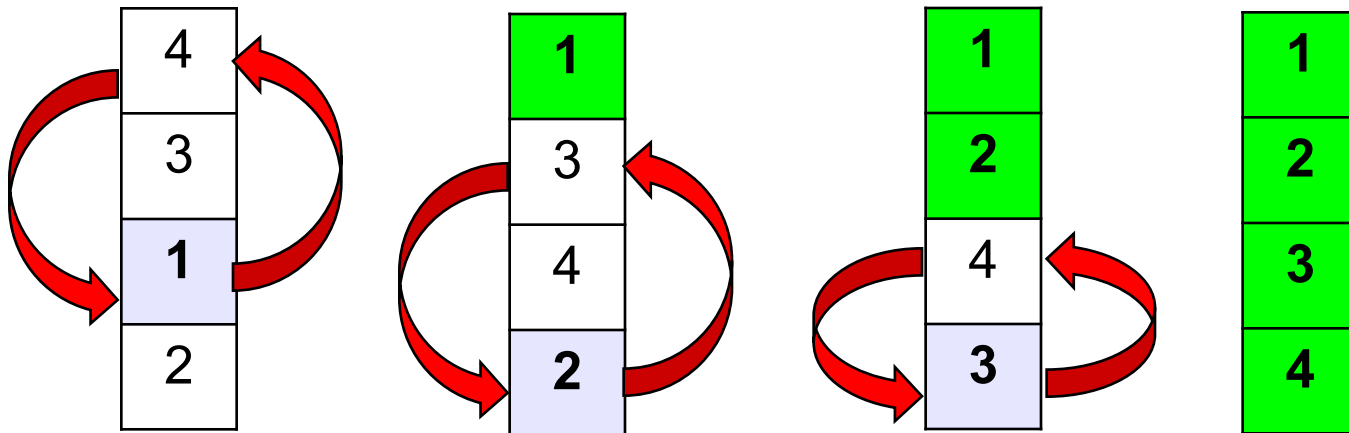
Метод бульбашки з прапорцем

```
i = 0;
do {
    flag = 0; // скинути прапорець
    for ( j = N-2; j >= i; j -- )
        if ( A[j] > A[j+1] ) {
            c = A[j];
            A[j] = A[j+1];
            A[j+1] = c;
            flag = 1; // підняти прапорець
        }
    i ++;
}
while ( flag ); // вихід якщо flag = 0
```

Метод вибірки

Ідея :

- Знайти мінімальний елемент і поставити його на місце (поміняти місцями з $A[0]$)
- Із **решти** елементів знайти мінімальний і поставити його на друге місце (поміняти місцями з $A[1]$), і т.д.



Метод вибірки

Потрібно $N-1$ проходів

```

for ( i = 0; i < N-1; i++ ) {
    nMin = i;
    for ( j = i+1; j < N; j++ )
        if ( A[j] < A[nMin] ) nMin = j;
    if ( nMin != i ) {
        c = A[i];
        A[i] = A[nMin];
        A[nMin] = c;
    }
}

```

Пошук мінімального
від $A[i]$ до $A[N-1]$

Якщо потрібно,
переставляємо



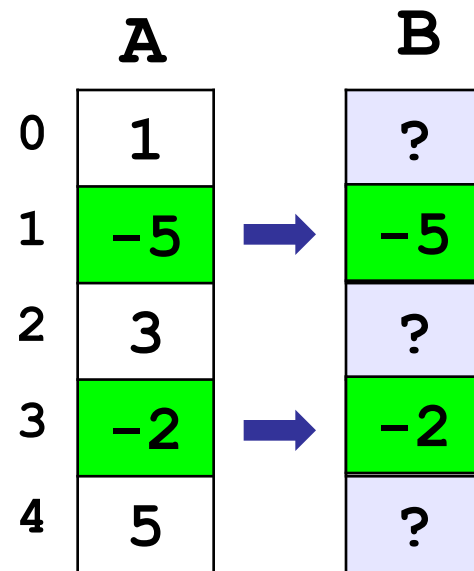
Чи можна забрати `if`?

Формування масиву за умовою

Завдання – знайти в масиві елементи які задовільняють деяку умову (наприклад, від'ємні), і скопіювати в інший масив.

Елементарне рішення:

```
const int N = 5;
int A[N], B[N];
// тут заповнити масив A
for( i=0; i < N; i++ )
    if( A[i] < 0 ) B[i] = A[i];
```

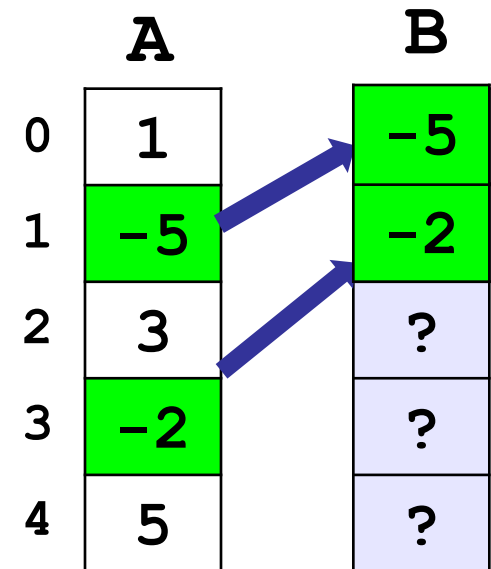


- Вибрані елементи не поруч, не в початку масива
- Незрозуміло як з ними працювати

Формування масиву за умовою

Рішення: ввести лічильник знайдених елементів `count`, наступний елемент ставиться на місце `B[count]`.

```
int A[N], B[N], count = 0;
// тут заповнюється масив A
for(i=0; i < N; i++)
    if(A[i] < 0) {
        B[count] = A[i];
        count++;
    }
// вивід масива B
for(i=0; i < count; i++)
    printf("%d\n", B[i]);
```



Програмування мовою C++

Тема. Пошук в масиві

Пошук в масиві

Завдання – знайти в масиві елемент, рівний **X**, або встановити, що він відсутній.

Рішення: для довільного масиву: **лінійний пошук**
недолік: **низька швидкість**

Як прискорити? – завчасно підготувати масив до пошуку

- Як саме підготувати?
- Як використовувати «підготовлений» масив?

Лінійний пошук

nX – номер
потрібного елемента
в масиві

```
nX = -1; // доки не знайшли.
for ( i = 0; i < N; i ++ ) // цикл по всіх елементам
    if ( A[i] == X ) // якщо знайшли, тоді...
        nX = i; // ...запам'ятовуємо номер
if ( nX < 0 ) cout << "Не знайшли... "/n;
else cout << "A[" << nX << "]=" << X/n;
```

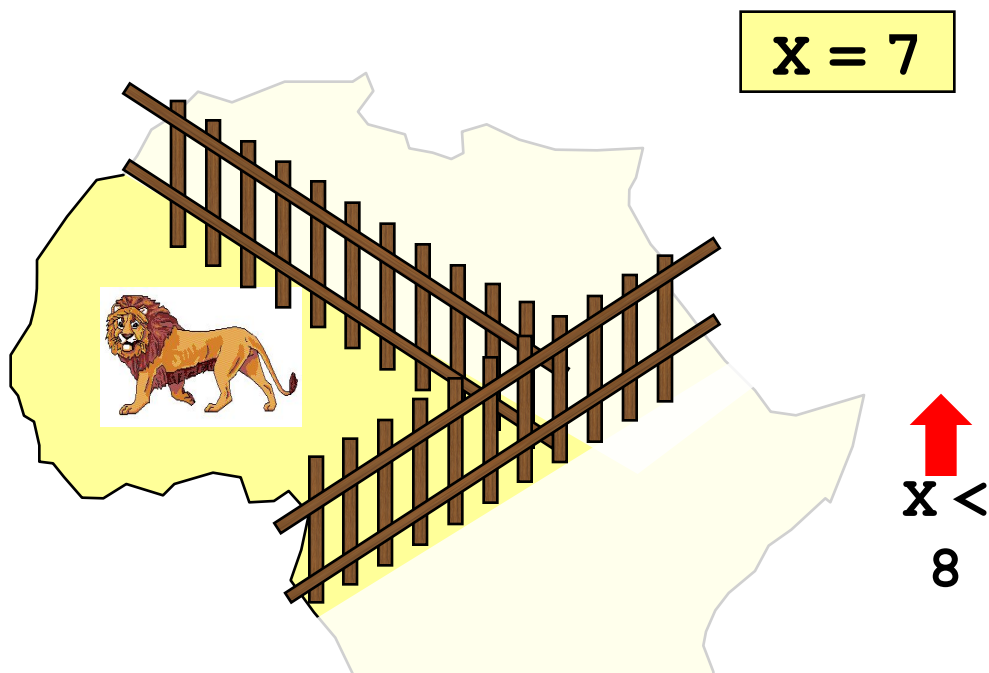


Що можна покращити?

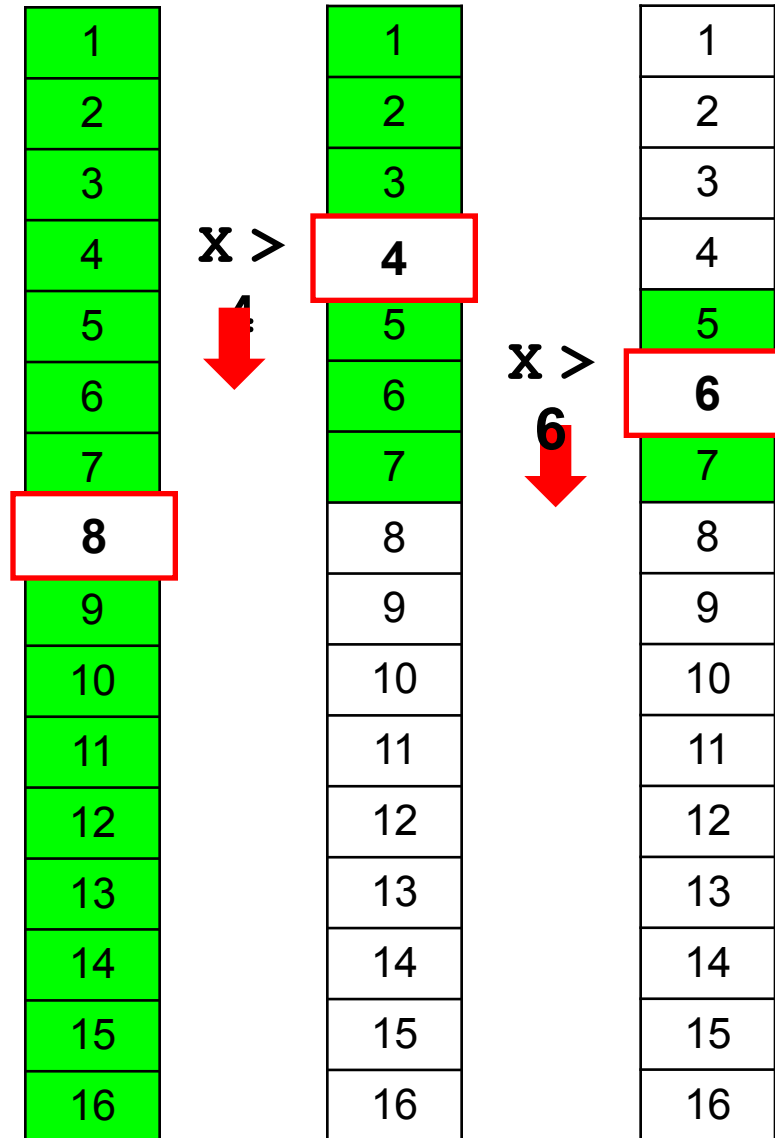
Поліпшення: після того як знайшли X, виходимо з циклу.

```
nX = -1;
for ( i = 0; i < N; i ++ )
    if ( A[i] == X ) {
        nX = i;
        break //вихід з циклу
    }
```

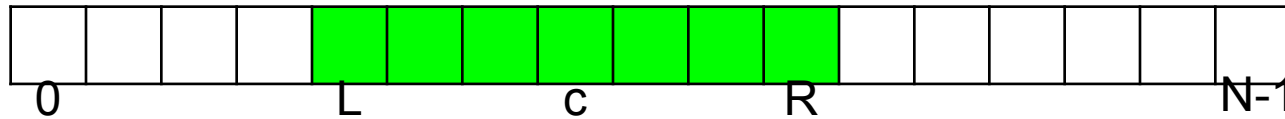

Двійковий пошук



1. Вибираємо середній елемент $A[s]$ і порівнюємо з X .
2. Якщо $X = A[s]$, знайшли (вихід).
3. Якщо $X < A[s]$, шукати далі в першій половині.
4. Якщо $X > A[s]$, шукати далі в другій половині.



Двійковий пошук



```
nX = -1;
L = 0; R = N-1; // границі : від A[0] до A[N-1]
```

```
while ( R >= L ){
    c = (R + L) / 2;
    if (X == A[c]) {
        nX = c;
        break;
    }
```

Номер середнього елемента

Якщо знайшли...

Вийти із циклу

```
    if (X < A[c]) R = c - 1;
    if (X > A[c]) L = c + 1;
}
```

Здвигаємо
границі

```
if (nX < 0) cout << "Не знайшли... "/n;
else      cout << "A[" << nX << "]= " << X/n;
```



Чому неможна `while (R > L) { ... } ?`

Порівняння методів пошуку

	Лінійний	Двійковий
підготовка	Не має	відсортувати
	Кількість кроків	
N = 2	2	2
N = 16	16	5
N = 1024	1024	11
N = 1048576	1048576	21
N	$\leq N$	$\leq \log_2 N + 1$