Покажчики та масиви

*

Дивись у корінь.
Козьма Прутков

Покажчики

- Покажчики (вказівники) призначені для збереження адрес областей пам`яті й дозволяють маніпулювати об`єктом, що розташований в цій пам`яті. В C++ розрізняють *покажчики: на об`єкт, на функцію, на void*. Зараз мова буде йти про покажчики на об`єкт (змінні).
- Масиви та покажчики в C++ дуже тісно пов`язані. Покажчики дозволяють виконати довільну операцію з масивами. Ім`я масиву розглядається як константний покажчик на початок відповідної області пам`яті.
- Покажчики використовуються в C++ дуже інтенсивно й не тільки для роботи з масивами.
- Одне з основних застосувань покажчиків полягає в роботі з динамічно створеними об`єктами: динамічними масивами, списками, деревами.
- Покажчики дозволяють передавати в функції “великі” об`єкти.

Покажчики

Кожний покажчик асоціюється з деяким типом даних. Область пам`яті, що адресується покажчиком, інтерпретується як значення відповідного типу.

Визначення покажчика має вигляд:

<тип> *<ім`я змінної>;

Наприклад:

int *a, b, *c;

Покажчик може бути змінною, або константою, а також вказувати на змінну, або константу.

Приклади

```
int i;  //ціла змінна
const int ci=1;  //ціла константа
int *pi;  //покажчик на цілу змінну
const int *pci;   //покажчик на цілу константу
int *const cp = &i;  //покажчик-константа на цілу
                     //змінну
const int *const cpc=&ci; //покажчик-константа на цілу
                          //константу
```

Ініціалізація покажчиків

При визначенні покажчика бажано здійснити його ініціалізацію. Способи ініціалізації:

1. Присвоювання адреси існуючого об`єкта:
   - операцією &

     ```
     int a = 5;  //ціла змінна a
     int *p = &a; //покажчик - адреса змінної a
     int *p (&a);  //теж саме, інший спосіб
     ```
   - іншим ініційованим покажчиком

     ```
     int *r = p;  //p отримав значення
     ```
   - за допомогою масиву

     ```
     int b[10]; //масив
     int *p = b;  //присвоювання адреси початку масиву
     ```

Ініціалізація покажчиків

2. Присвоювання адреси пам`яті в явному вигляді:
   char *p = (char *)0xB80000000;
3. Присвоювання порожньої адреси:
   int *pnt = NULL;
   int *pp = 0;
4. Виділенням динамічної пам`яті:
   int *pnt = new int;
   int *pp = new int (10);
   int *qq = new int [10];
   вимагає явного звільнення пам`яті за допомогою операції delete
   delete pnt; delete [] qq;

Покажчики

Можна описувати складні типи. Діють правила:

- За означенням (), [] мають однаковий пріоритет більший за пріоритет *, розглядаються зліва-направо.
- Якщо праворуч від імені [] – це масив, якщо () – функція.
- Якщо ліворуч від імені * - це покажчик на проінтерпретовану раніше конструкцію.
- Якщо праворуч) – потрібно застосувати правила для внутрішньої частини () , а потім переходити до зовнішньої частини.
- В останню чергу інтерпретується специфікатор типу.

Наприклад: `int *(*p[10])();` // ???

Операції з покажчиками

- * - розіменування, & - отримання адреси,
  new – виділення та delete – звільнення пам`яті:
  *pi = abs(*pi);
- присвоювання: pi = pnt;
- порівняння: pi == pnt
- арифметичні мають сенс при роботі з структурами послідовно розташованими в пам`яті (наприклад - масиви) :
  - додавання константи (pi + 5)
  - віднімання константи (pi - 2)
  - різниця покажчиків (pi - ri)
  - інкременту (++) (pi++) (++pi)
  - декременту (--) (pi--) (--pi)
  - фактично дії відбуваються з одиницями виміру sizeof(<тип>).

Масиви та покажчики

В результаті визначення масиву у змінній зберігається адреса його першого елементу. Ім`я змінної-масиву – покажчик на перший елемент. Тому звернення до елементів можливі як за індексами, так й за результатами “адресної арифметики”.

Наприклад:

```
int arr[3] = {1, 2, 3};
```

наступні вирази еквівалентні:

```
arr[1]
*(arr + 1)
*(1 + arr)
1[arr]
```

Масиви та покажчики

```
const short size = 3;
int *p = 0;  int arr[size] = {1, 2 ,3};
p = arr;    p = &arr[0];  //еквівалентні

//перебір елементів
for (int i=0; i<size; i++) {
cout << *p << endl;  ++p;  }

//теж перебір елементів
for (int *q=arr; q<arr+size; ++q) {
cout << *q << endl; }

//теж перебір елементів
p = arr; i = size; while (i-- > 0) {cout << *p++;}
```

Масиви та покажчики

```
//копіювання масиву
const int k = 100;
void copy_arr(int n, double a[k], double b[k])
 {int *pa = a;
  int *pb = b;
  for (; n>0; n--)  *pb++ = *pa++;
 }
//звернення
double aa[k];
double bb[k];
...
copy_arr(k,aa, bb);
```

Приклад

```
//бінарний пошук
int bin_search(int key, const int *arr, int count) {
 if (count < 1 || !arr) return -1;
 int beg = 0, end = count - 1, i = 0;
 while (beg <= end) {
   i = (beg + end) / 2;
   if (arr[i] == key) return i;
   if (arr[i] < key) beg = i + 1;
   else end = i - 1;
  }
 return -1;
}
```

Масиви покажчиків

Можна визначати масиви покажчиків:
<тип> *<ім`я масиву> [<кількість елементів>];

Наприклад:

```
int *p[3];
int x=10, y=20, z=30;
p[0] = &x;
p[1] = &y;
p[2] = &z;
cout << *p[0] << "  " << *p[1] << "  " << *p[2] << endl;
```

Динамічні масиви

Попередні масиви – “статичні масиви”. Кількість елементів задавалась при визначенні масиву (до виконання програми) й не підлягала змінам.

Є можливість динамічного виділення пам`яті:
<покажчик> = new <тип>[<кількість елементів>];

Звільнення пам`яті:
delete [] <покажчик>;

При звільненні пам`яті кількість елементів не вказується.

Відповідальним за повернення пам`яті є програміст.

Приклад

```
int main(){
 int n;
 int *p = 0,  *q = 0;
 cout << "n= ";   cin >> n;  cout << endl;
 p = new int [n];   q = p;
 cout << "array A: ";
 for (int i=0; i<n; i++)   cin >> *q++;
 cout << endl;     //q==?
 q = p;
 //виведення даних
 cout << "array A: ";
 for (int i=0; i<n; i++)  cout << *q++ << "  " ;
 delete [] p;  p = 0;
return 0; }
```

Динамічні масиви

Потрібно враховувати, що при виділенні пам`яті може виникнути ситуація неможливості надати заявлений обсяг пам`яті.

В сучасному стандарті C++ активується виключення bad_alloc , що визначено у файлі <new>.

Приклад

Багатовимірні динамічні масиви

При створенні в операції new потрібно вказувати всі виміри (лівий вимір може бути змінною).

Наприклад:

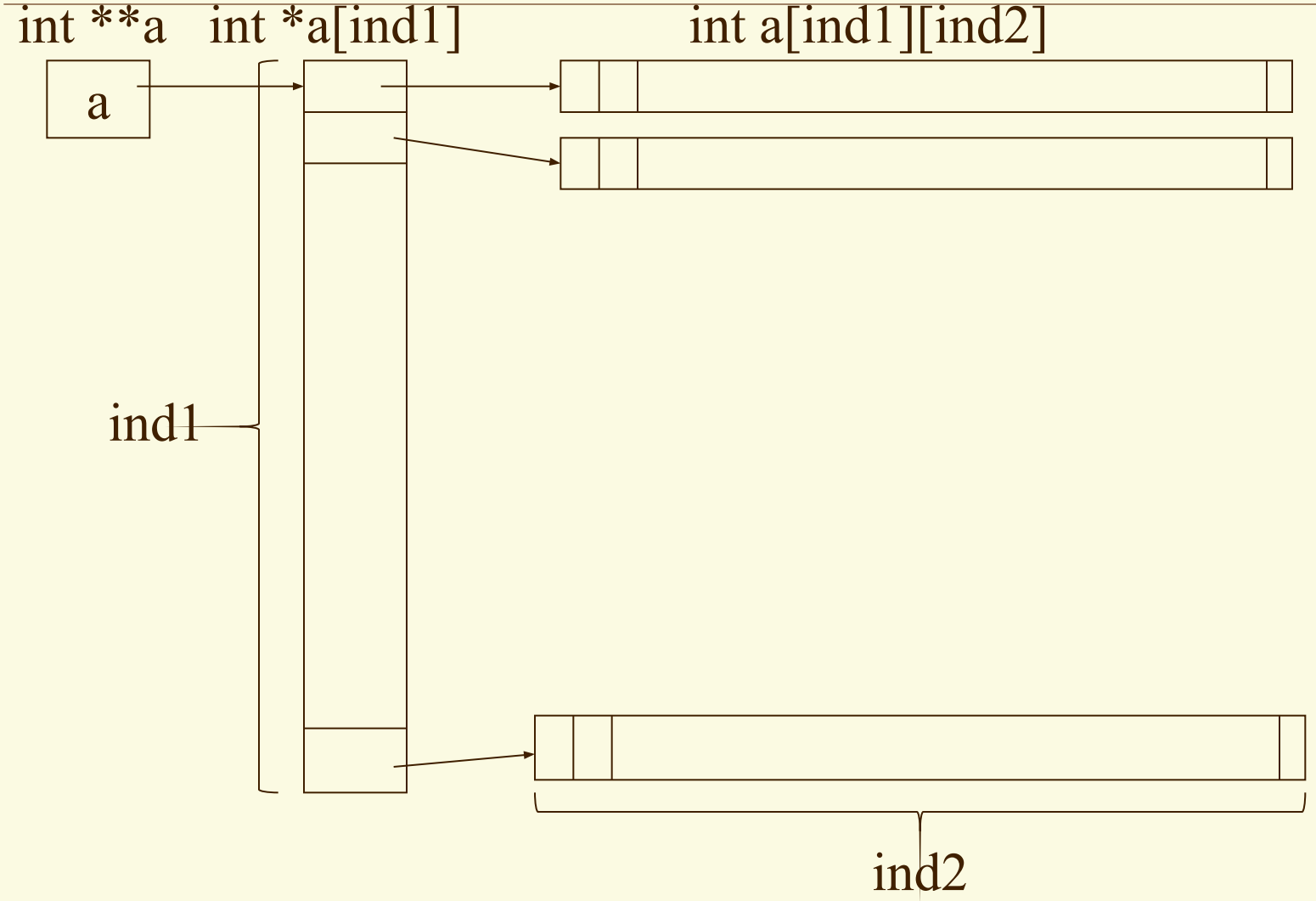
```
int ind1 = 5;
int **parr = (int **) new int [ind1][10];
```

Багатовимірні динамічні масиви

Більш безпечний та універсальний спосіб виділення пам`яті, без вказаного обмеження на виміри:

```
int ind1, ind2;
cout << "Size array n*m :";
cin >> ind1 >> ind2;
int **a = new int *[ind1];
for (int i = 0; i < ind2; i++)
  a[i] = new int[ind2];
```

Багатовимірні динамічні масиви



Зауваження

- Дотримуватись розглянутих правил використання спефицікаторів при визначенні покажчиків та масивів покажчиків (наприклад, int *p[10] – масив з 10 покажчиків).
- Використовуючи явні механізми для виділення пам`яті на забувати про необхідність звільнення виділеної пам`яті. Враховувати, що при звільненні відповідний покажчик не змінюю свого значення.
- При виділення пам`яті враховувати можливість відсутності потрібного обсягу вільної пам`яті.
- Не порушувати правил “адресної арифметики” та порівнянь покажчиків.
- Не виходити покажчику за межі пам`яті об`єкту.
- Не допускати ситуацій з “втраченою пам`яттю”.

Підсумки

- Розглянули лише покажчики на змінні, їх зв`язок з масивами, використання покажчиків для доступу до елементів масивів.
- Використання покажчиків дозволяє отримувати більш гнучкі та ефективні рішення, але при нехтуванні правилами може бути джерелом чималих проблем.
- Можливості використання покажчиків не вичерпуються лише розглянутими питаннями.
- Бібліотеки С надають також інші можливості для явного керування пам'яттю.

Задачі
(використовуючи розглянуті можливості)

- Пошук заданого значення у масиві з цілих:
  - масив одновимірний;
  - масив одновимірний впорядкований;
  - масив двовимірний;
  - масив двовимірний впорядкований;
- Перевірити, чи є задане число паліндромом.
- Записати функцію для “бульбашкового” сортування масиву.
- Записати функцію для визначення чи є квадратна матриця симетричною.
- Записати функцію для транспонування квадратної матриці.

Задачі

- Для заданої дійсної матриці знайти індекси всіх її “сідлових точок” (елементи, що є одночасно найменшими у рядку й найбільшими у стовпчику, або навпаки.)
- Визначити чи є ціла квадратна матриця “магічним квадратом” (суми елементів у всіх рядках та стовпчиках однакові).
- В місті М діє p-ічна система числення, а номери тролейбусних квитків містять 2k розрядів. Квиток вважається щасливим, якщо сума перших k розрядів дорівнює сумі останніх k розрядів.

  Вхід: Значення p та k.

  Вихід: Кількість щасливих квитків.

Задачі

- Обчислити масив A[100], який містить 100 перших елементів (у зростаючому порядку) множини M, що визначається наступним чином:
  - 1. 1 належить M;
  - 2. якщо x належить M, то y=2*x+1 та z=3*x+1 належать M;
  - 3. ніякі інші числа не належать M.
- Побудувати перші N натуральних чисел дільниками яких є тільки числа 2, 3, 5.
- Задані A, B, N. Знайти всі натуральні числа, що не перевищують N, які представляються у вигляді суми довільної кількості доданків, кожен з яких A або B.