Бублик Володимир Васильович

Процедурне програмування C/C++

Лекція 5. Базові поняття програмування. Указники і відсилки

консультації вівторок, 15 год.
кімн. 1/331

Лекції для студентів 2 курсу

Указники (Pointers)

- Спосіб, у який С оперував указниками, був блискучою інновацією, він вирішив багато проблем, які до цього були в структуруванні даних, і забезпечив гарний вигляд програмам на майбутнє.
  Donald Knuth
- Для того щоб бути хорошим програмістом, важливо розуміти, що насправді відбувається "за кулісами" високорівневої мови програмування.
  Alexander Stepanov, розробник STL

Призначення указників

Указник (pointer) набуває значеннями адреси пам'яті, а також особливе нульове значення, з яким не зв'язана жодна адреса

Динамічне виділення і звільнення пам'яті:

- створення масивів;
- створення динамічних структур даних (списків, дерев, тощо)

Типізація указників

Кожному указнику приписано його тип:

- указник цілого;
- указник символу;
- указник дійсного, тощо.

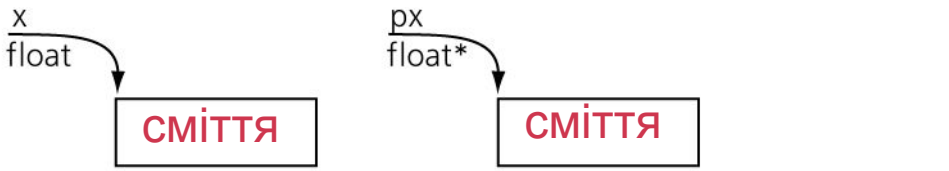
З одним указником можна зв'язати цілий агрегат даних одного й того ж типу, розміщених одне за одним, починаючи з місця, позначеного указником

Визначення указника

1. int *ptrI;
2. float *px, y, z; //Ось чому погано писати в ряд
3. double *px, *py, *pz;

Визначення без ініціалізації приводять до заповнення пам'яті сміттям. Засмічені указники (dangling pointer) небезпечні!

float x, *px;



некоректні дані некоректна адреса

Pointer vs. goto

Указники приводять до тих же проблем в структурах даних, до яких приводять оператори переходу в структурах керування.

Скрізь, де зуміємо, уникаємо указників. Якщо вживаємо, то дотримуємось суворої дисципліни!

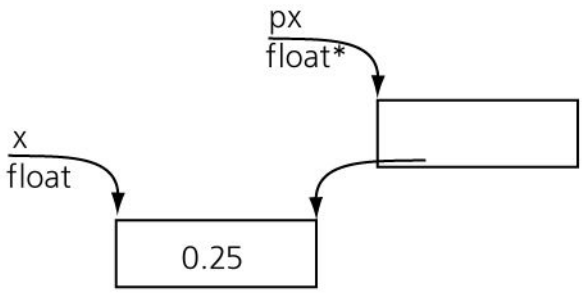
Якщо значення указника невідоме, ініціалізуємо його нулем

```
double *px = 0;
```

Якщо значення указника не нуль, то з ним зв'язані два елементи пам'яті!

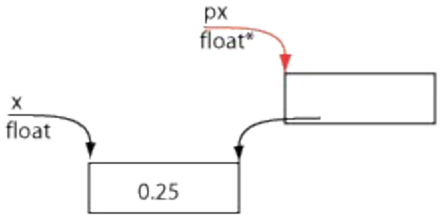
Операція адресування

- x=0.25; px = &x;

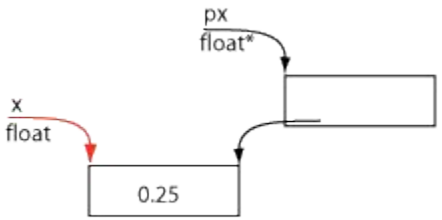


- & одномісна операція адресування

lvalue(px)

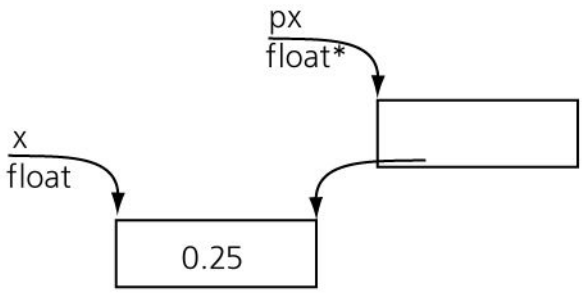


rvalue(px) == lvalue (x) == rvalue(&x)



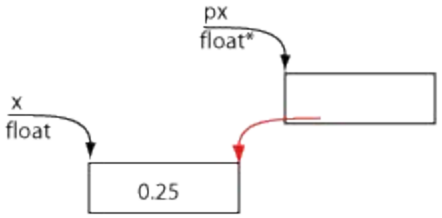
Операція розіменування

- cout<< *px;

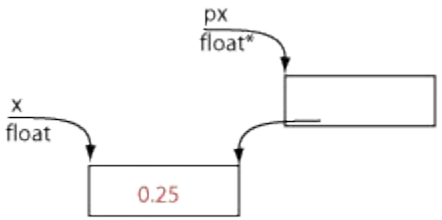


- * одномісна операція розіменування

rvalue(px) == lvalue(x)



rvalue(*px) == rvalue(x)



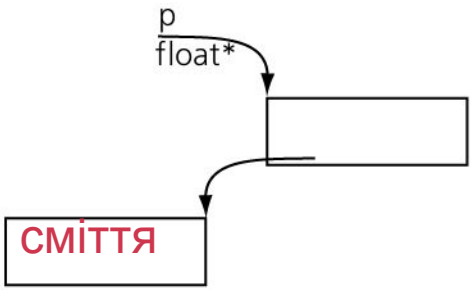
Адресування і розіменування

- Взаємна оберненість операцій над пам'яттю

- &(*px) == px
- px == *(&px)

Динамічне виділення пам'яті

- float *p = new float;
- new операція виділення нового елемента пам'яті, його адреса зберігається в p, значення *p заповнено сміттям



- px = new float;

Нестача пам'яті

Стандартна реакція: виникнення аварійної ситуації bad_alloc

- Обробка засобами системи програмування (аварійне припинення виконання програми)
- Програмна обробка переривання (оператор catch в блоці випробувань try – буде далі)

Замовна реакція: повернення нульового указника

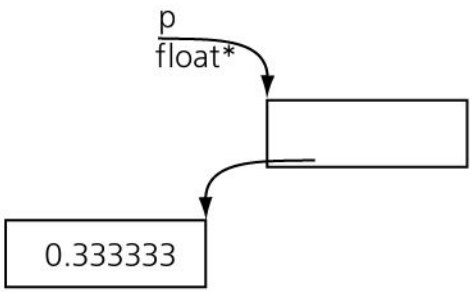
- Формат виклику new (nothrow)

Приклад

- double *px = new (nothrow) double [bigAmount];
- if (px==0) //Вільної пам'яті не вистачило

Динамічне виділення і ініціалізація пам'яті

- float *p = new float (0.333333);
- Тепер значення *p коректне



Правило гарного тону

- Якщо ви виділили динамічну пам’ять за допомогою команди new не забудьте своєчасно звільнити її командою delete та обнулити указник

```
float *p = new float (0.333333);
// робіть все, що вам потрібно
delete p;
// потурбуйтесь про захист від завислих указників
p = 0;
```

Звільнення памяті

- double *pd = new double (5.2);
- cout<<pd<<endl; // 0x004419B0
- cout<<*pd<<endl; //5.2
- delete pd;
- cout<<pd<<endl; // 0x004419B0
- cout<<*pd<<endl; //-1.45682e+144

- double *new_pd = new double;
- cout<<new_pd<<endl; // 0x004419B0
- cout<<*pd<<endl; //-6.27744t+066
- cout<<*new_pd<<endl;

Звільнення памяті

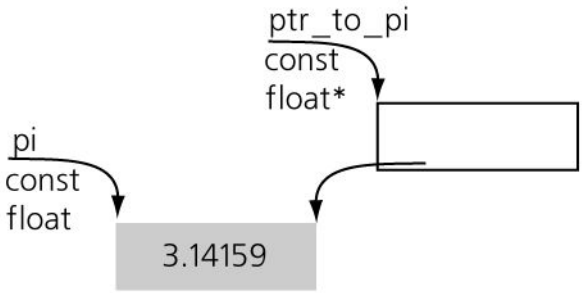
```
double *pd = new double (5.2);
cout<<pd<<endl;        // 0x004419B0
cout<<*pd<<endl;       //5.2
delete pd;
//    Правило гарного тону
pd = 0;
```

Висновок про управління пам'яттю

- Кожному new свій delete
- Перевіряйте наявність вільної пам'яті
- Слідкуйте за тривалістю життя динамічних об'єктів у пам'яті

Сталі величини і указники

- *Указник сталої*
- const float pi = 3.14159;
- const float *piPtr = &pi ;
- //*piPtr = 4; ERROR!
- // float *piNonConstPtr =π теж ERROR



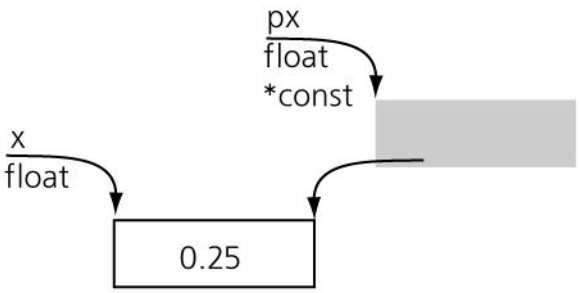
Права указника сталої

- float x = 1;
- const float* px = &x;
- x =3;
- cout<<*px<<endl;
- //*px = 4; як і раніше ERROR!

Указнику сталої не надано права змінювати об'єкт, але безпосередня зміна об'єкту, якщо він не сталий, і надалі можлива

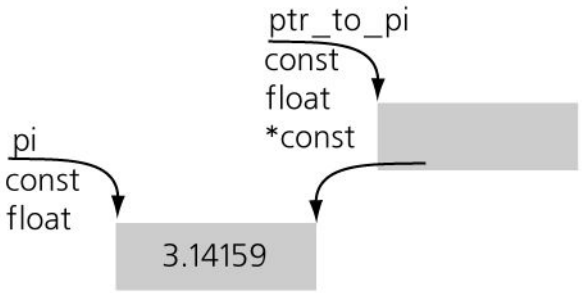
Сталі величини і указники

- *Сталий указник*
- float x = 0.25;
- float *const px = &x;
- // Сталий указник не можна перемістити
- // px = &y; ERROR!



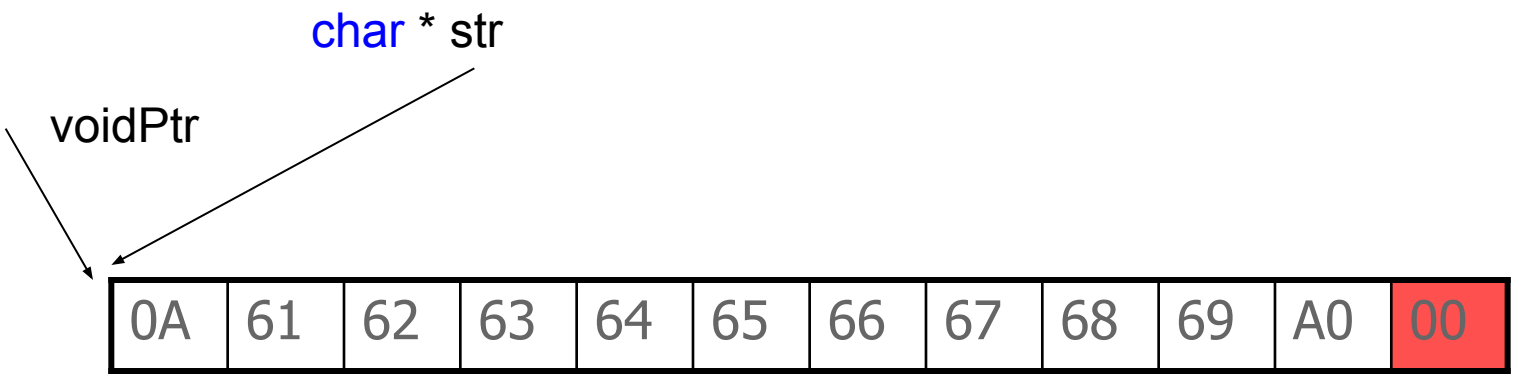
Сталі величини і указники

- *Сталий указник **сталої***
- const float pi = 3.14159;
- **const float** *const piPtr = π
- //*piPtr = 4; ERROR!
- //piPtr = &nAvogadro; ERROR!



Безтипові указники

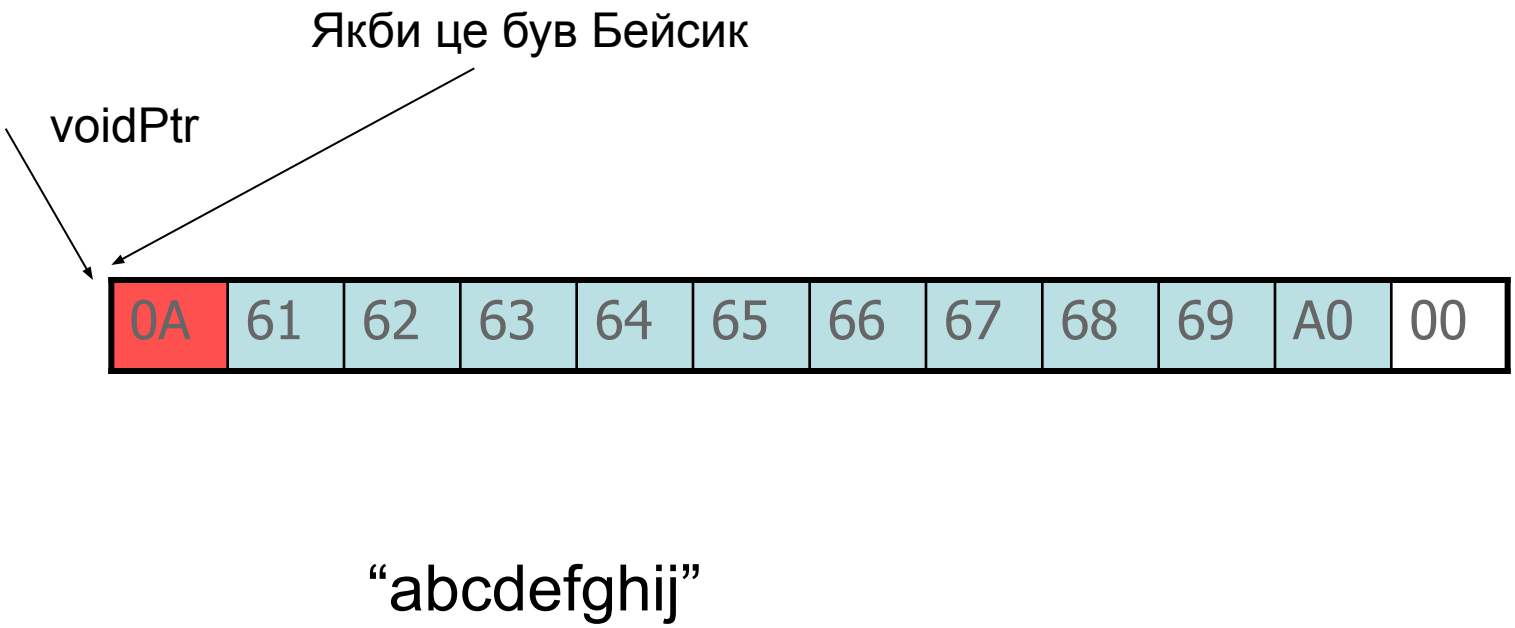
- void *voidPtr; //Що б це значило?



“\nabcdefghij”

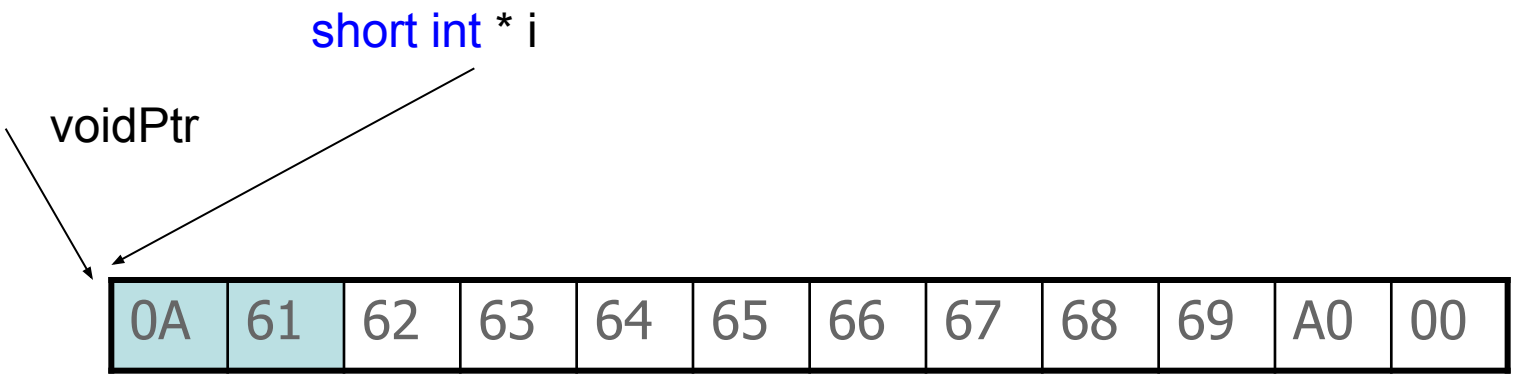
Безтипові указники

- void *voidPtr; //Що б це значило?



Безтипові указники

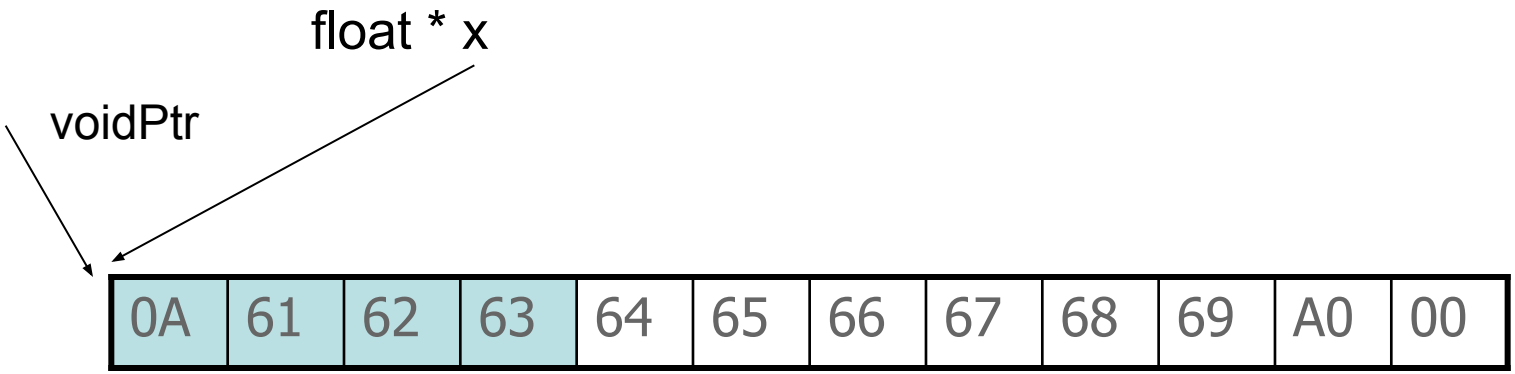
- void *voidPtr; //Що б це значило?



24842

Безтипові указники

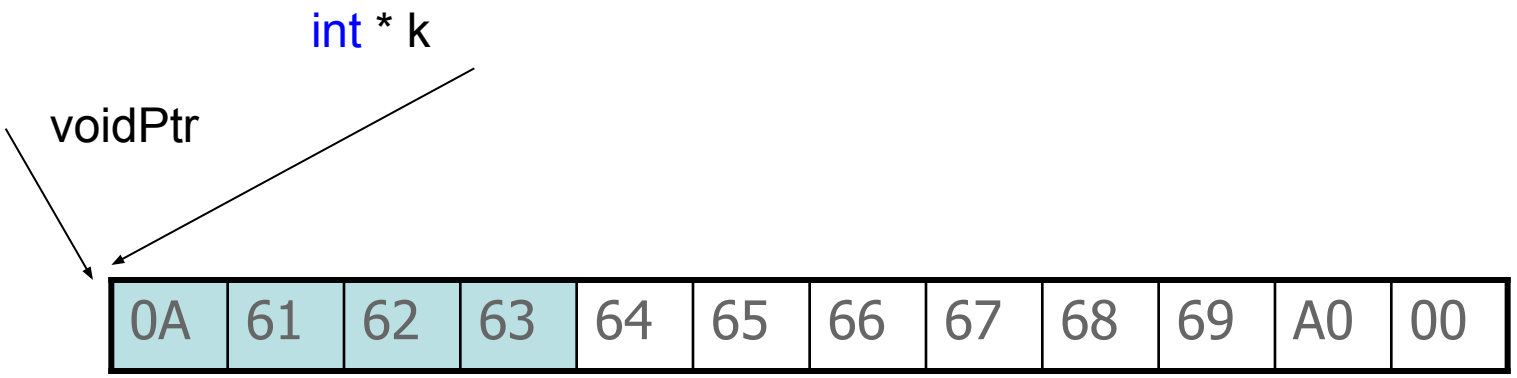
- void *voidPtr; //Що б це значило?



4.17596e+021

Безтипові указники

- void *voidPtr; //Що б це значило?



1667391754

Безтипові указники

- void *voidPtr; //Що б це значило?
- float *pf = new float (3.1);
- double *pd = new double (5.2);

- voidPtr = pf;
- cout<<voidPtr<<endl;
- // 0x004419F0
- // cout<<*voidPtr<<endl; розіменування неможливе
- voidPtr = pd;
- cout<<voidPtr<<endl;
- // 0x004419B0

Символьний указник

Поганий указник

- char *c = new char;
- cout<<*c<<endl;

Ініціалізований указник

- char *cc = new char ('a');
- cout<<*cc<<endl;

Особливість ініціалізації рядків

- // char *str = new char(“String?“);
- char *str = "Operation New is not needed";
- cout<<str<<endl;

Операції над указниками

```
float *p1= new float (1), *p2= new float (2);
cout<<p2;    //0x004418F0
```

1. Порівняння на рівність або нерівність
   ```
   p1 == p2; p1 > p2;…
   if(p2>p1) cout<<"p2>p1"<<endl;
       else cout<<"p1>p2"<<endl;
   ```
2. Присвоєння і ініціалізація
   ```
   p1 = p2; p1 =&x; p1 = new float (3);
   char *pc = new char [100];
   ```
3. Збільшення, зменшення
   ```
   p1++; ++p1; p1--;--p1; p1+10; p1-10;
   cout<<p2+1;    //0x004418F4
   ```

Куди рухатимемося (МООП)?

```
// Спектр інтелектуальних указників
template <typename Pointee>
class SmartPtr;
// Інтелектуальні указники масивів
template<typename Pointee>
class SmartPtr<Pointee[]>;
// Вкладені інтелектуальні указники
template<typename Pointee>
class SmartPtr<SmartPtr<Pointee>>;
```

Відсилка (псевдонім) (reference)

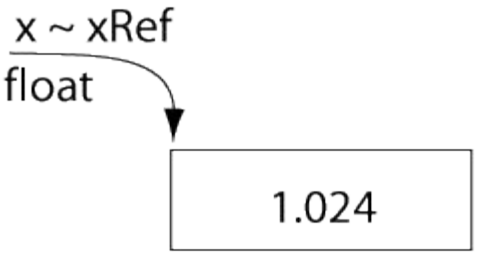
Відсилка або псевдонім — це альтернативне ім’я об’єкта, яке позначає об’єкт на рівних правах з його основним іменем.

При створенні псевдонім зв'язується зі своїм об'єктом (відсилає до нього) і ця відсилка дійсна протягом усього життя псевдоніму.

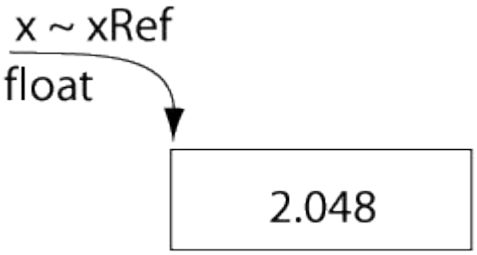
Для чого? Для того, щоб у різних частинах програми іменувати одну й ту ж область пам'яті зручним і зрозумілим для цієї частини іменем

Відсилка (псевдонім) (reference)

- float x = 1.024; //визначення змінної x
- float &xRef = x; //визначення її псевдоніму

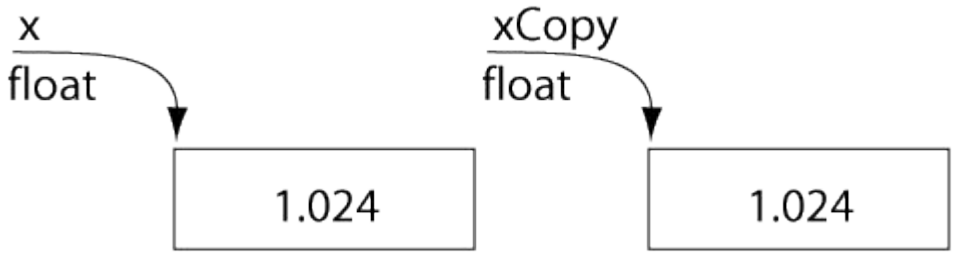


- xRef *= 2;// x == xRef == 2.048

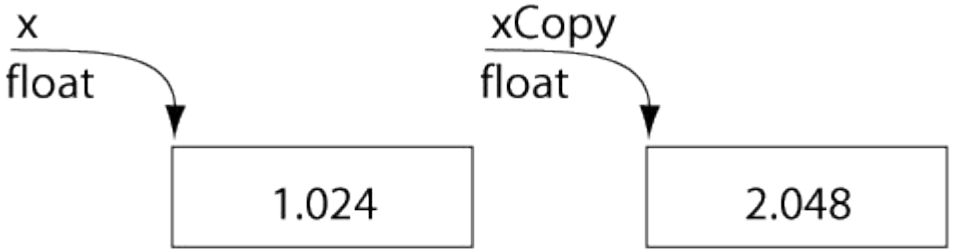


Копіювання

- float x = 1,024; //визначення змінної x
- float xCopy = x; //визначення її копії



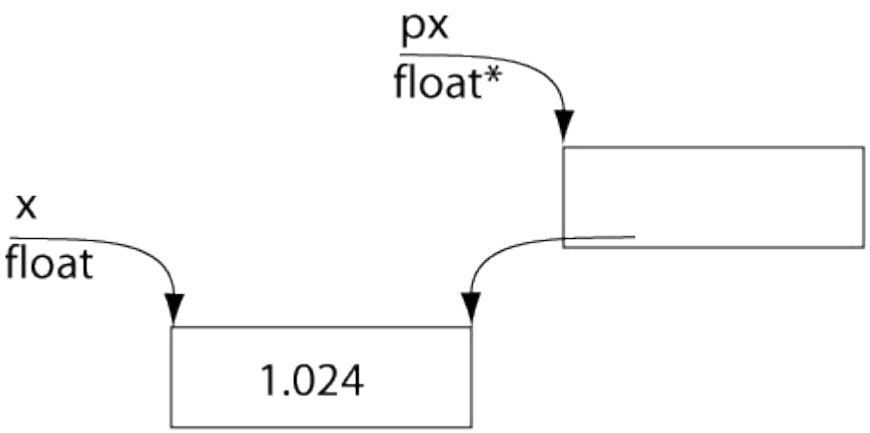
- xCopy += x; //подвоєння xCopy, x незмінний



Порівняння відсилок і указників

Визначення указника

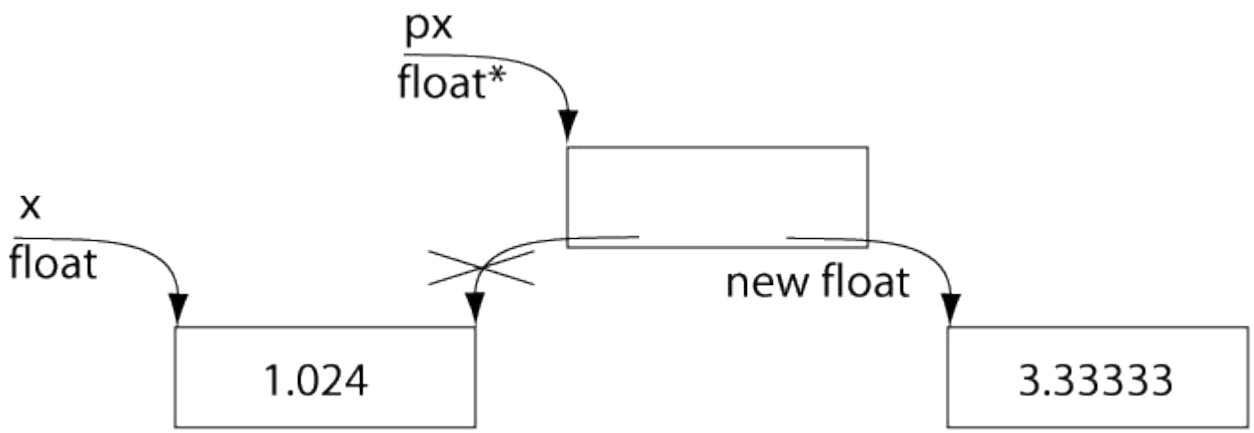
- float x = 1,024; //визначення змінної x
- float *px = &x; //визначення указника на неї



Порівняння відсилок і указників

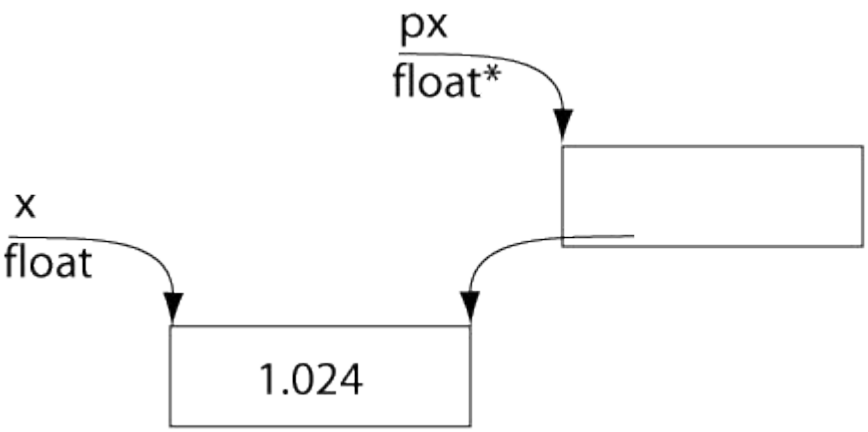
Перенаправлення указника

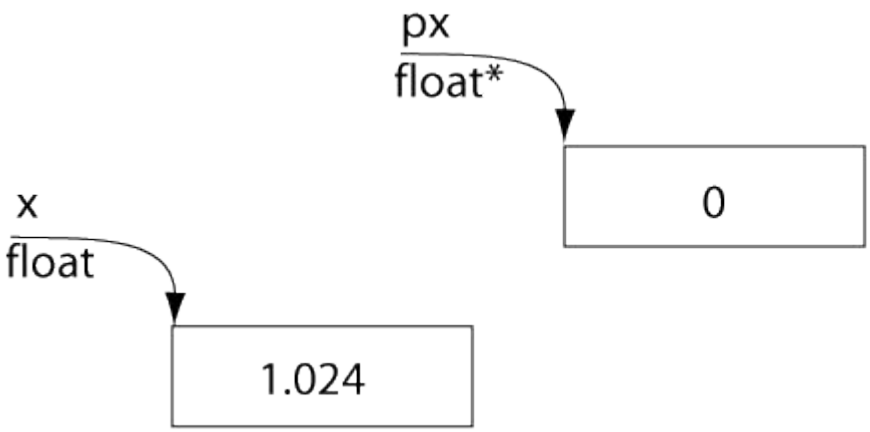
- px = new float (3.33333);
- // x == 1,024; *px == 3.33333
- //Зв’язок між px і x розірвано



Порівняння відсилок і указників

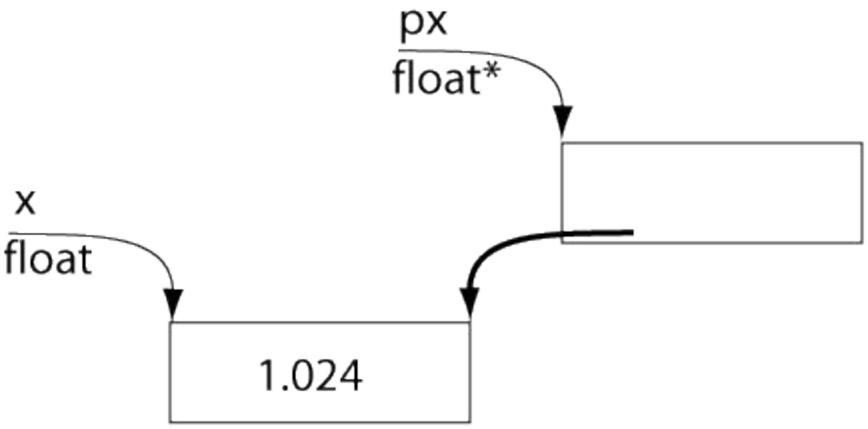
- Указнику p відповідає два елементи пам'яті, кожен з яких має власне значення.
- Зміна значення p розриває наявний зв’язок між елементами пам'яті, наприклад, px = 0;





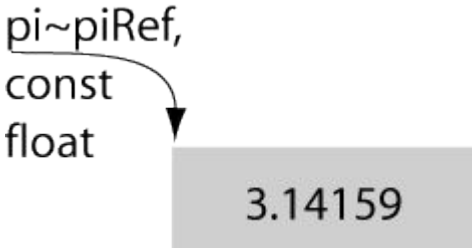
Відсилка і сталий указник

- float x = 1.024; //визначення змінної x
- float *const px = &x; //сталий указник на неї
- //схожість до відсилки: зв’язок нерозривний
- //відмінність: наявність двох елементів пам'яті



Стала відсилка

- const float pi = 3.14159;
- const float &piRef = pi;
- // float &piNonConstRef = pi; ERROR



Стала відсилка

Що б це значило?

- float x = 1;
- const float & rx = x;
- x =2;
- cout<<rx<<endl;

Якщо псевдонім сталий, то він не має права змінити об'єкт, але безпосередня зміна об'єкту, якщо він не сталий можлива

Указники указників

- double **ppd = new double*;
- *ppd = new double (5);
- cout<<ppd<<endl;
- cout<<*ppd<<endl;
- cout<<**ppd<<endl;

Використання указників на указники — розповсюджена техніка програмування у “чистому” С. Проблеми управління пам'яттю при цьому зростають, бо необхідно слідкувати за обома поверхами указника. Вживати ще обережніше, ніж звичайні.

Висновки

- Крім безпосередніх операцій з пам'яттю указники використовуються для передачі змінних параметрів при програмуванні на “чистому” С, а указники другого рівня для передачі змінних указників
- В сучасному програмуванні на С++ замість передачі указників використовується передача змінних параметрів відсилками

Домашнє завдання

- Розділ 2, стор. 51-71