Указатели и псевдонимы

Тип указател

- МС: всички възможни lvalue от даден тип и специалната стойност NULL
- Интегрален нечислов тип
- Параметризиран тип: ако T е произволен тип, T* е тип “указател към T”
- Физическо представяне: цяло число с размера на машинната дума, указващо адреса на реферираната lvalue в паметта

Операции с указатели

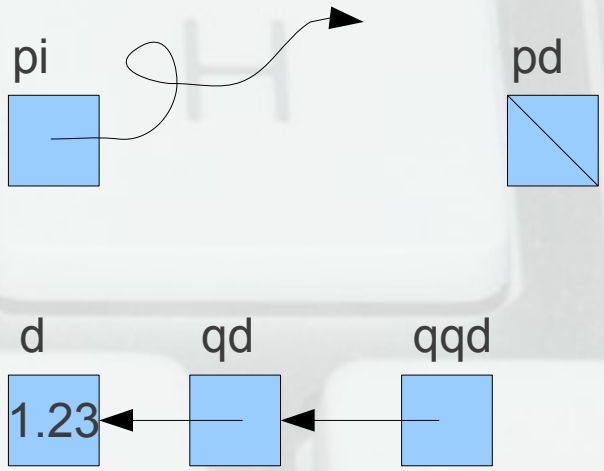
- реферииране (&<lvalue>)
- дереферииране (*<указател>) - унарен!
- указателна аритметика (+, -, +=, -= ++, --)
- сравнение (==, !=, <, >, <=, >=)
- извеждане (<<)
- няма въвеждане (>>) !

Дефиниране на указателни променливи

- <тип> *<идентификатор> [= <израз>] {, *<идентификатор> [= <израз>] };

Примери:

```
int *pi;
double *pd = NULL;
double d = 1.23;
double *qd = &d;
double **qqd = &qd;
```



Реферирane и дереферирane

- int x = 5, *p = &x, *q = p, y = *p + 2;
- *p++; p = &y; *q = 1; *p = *q;
- Внимание:
  - &<lvalue> е rvalue! ~~(&x = p)~~
  - *<rvalue> е lvalue! (*p = x)
  - &(*p) == p
  - *(&x) == x

Указателна аритметика

- sizeof(<тип> | <израз>) — връща размера в байтове заеман в паметта от <израз> или променлива от <тип>
- <указател> [+|-] <цяло число>
  <цяло число> + <указател>
- T* p;
  p + i ↔ (T*)((int)p + i*sizeof(T))

Указател към неизвестен тип

- void*
- Преобразуване **без** загуба **от** произволен указателен тип

  int x, *p; void *q = p, *r = &x, *s = &r;
- Няма преобразуване **към** произволен указателен тип

  int* p; void* q = p; ~~int* r = q;~~ int* s = (int*)q;
- Няма дереферииране (тип void е празен)

  int x; void* p = &x; ~~*p = 2; void y = *p;~~

Указатели и константи

- Константен указател

  <тип> * const

  int x, *p = &x; int *const q = p; ~~q = p + 2;~~ *q = 5;

- Указател към константа

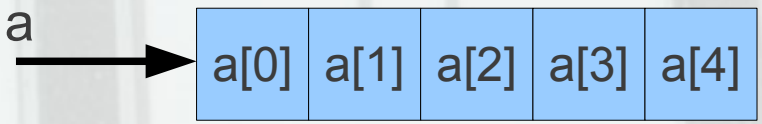
  const <тип>* ↔ <тип> const*

  int x, *p = &x; int const* q = &x; q++; ~~q = p; *q = 5;~~

- Ако p е указател към константа, то *p е <rvalue>
- Ако x е константа, то &x е указател към константа

Указатели и масиви

- Името на масив е константен указател към първия му елемент
- a[i] ↔ *(a+i)



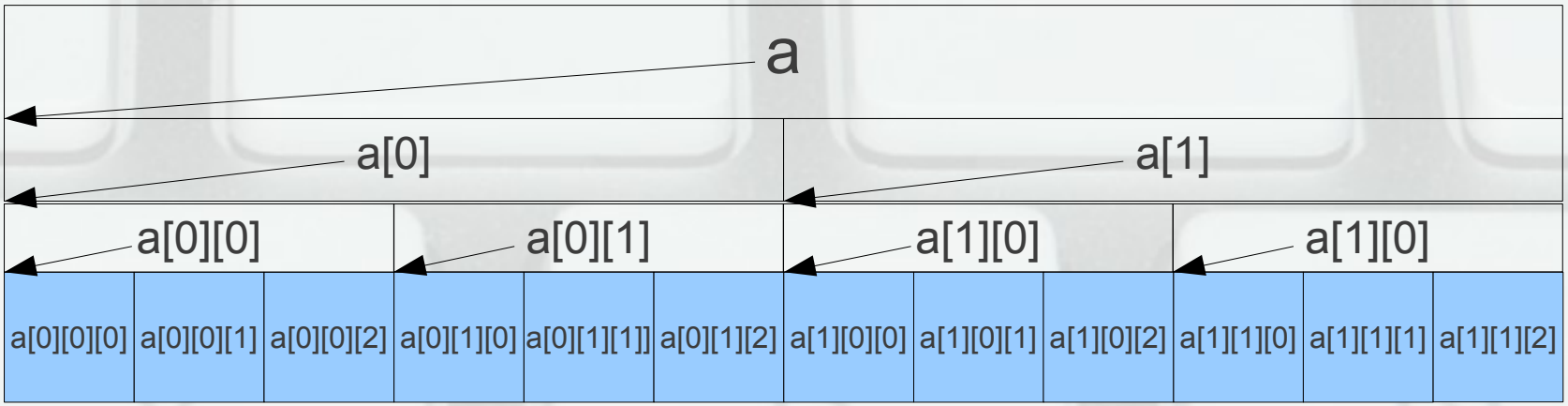

```
int a[5], x;
cout << *a; *(a+1) = 7; *(a+4)--;
a++; a--; a = &x;
```

- Странно, но вярно: a[i] ↔ *(a+i) ↔ *(i+a) ↔ i[a]

Указатели и многомерни масиви

- int a[2][2][3];
- a е от тип int* const [2][3];
- a[i] е от тип int* const [3];
- a[i][j] е от тип int* const;

- a[i] ↔ *(a+i)
- a[i][j] ↔ *(*(a+i)+j)
- a[i][j][k] ↔ *(*(*(a+i)+j)+k)
- a[1][1][1] ↔ *(*(*(a+1)+1)+1)



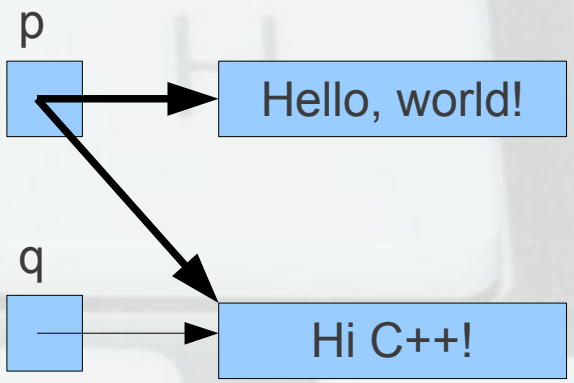
Указатели и низове

- Низовете са масиви от символи
- Името на низ може да се разглежда като константен указател от тип char * const

```
char s[] = “ Hello, world!”;
char* p = s;
while (*p) {
    cout << *p << ' ';
    p++;
}
```

Указатели и низови константи

- char const* p = “Hello, world!”;
- char* q = “Hi C++!”;
- p = q;
- ~~q[1] = 'o';~~
- cout << p[4];

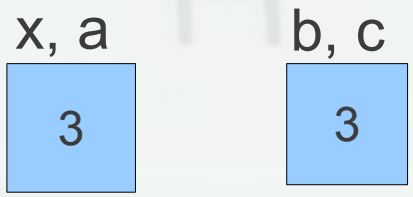


Псевдоним

- МС: всички възможни lvalue от даден тип
- Параметризиран тип: ако T е произволен тип, T& е тип “псевдоним на T”
- Физическо представяне:
  - не е специфицирано
  - де факто еквивалентно на константен указател към T

Дефиниране на псевдоним

- <тип>& <идентификатор> = <обект>
  {, &<идентификатор> = <обект>};
- int x = 3;
  int &a = x, b = a;
  int &c = b;
  a = c + 5; // ?



- Инициализацията е задължителна
- Псевдонимът не може да се пренасочва към друг обект

Свойства на псевдоними

- Самият псевдоним не може да бъде манипулиран
- Псевдонимът не се различава от оригинала
- Псевдонимите на един и същ обект са взаимнозаменяеми