Манипулатори, многомерни масиви, указатели

гл.ас. д-р. Нора Ангелова

Задача 1. Напишете 20 правилни израза, ако израз е дефиниран по следния начин:

<израз> ::= <цифра> | s(<израз>) | p(<израз>)

<цифра> ::= 0 | 1 | 2 | … | 8 | 9;

0 1 2 3 4 5 6 7 8 9

s(5) p(3)

s(p(3)) p(s(5))

p(p(p(s(4))) s(s(p(p(6)))) s(s(p(p(s(p(8))))))

p(p(s(s(s(8))))) s(p(s(p(8)))) s(s(s(p(1))))

s(s(s(s(p(p(p(p(p(9))))))))))

Задача 2. Намерете и обяснете грешките във фрагмента:

```
double x, y = 1.5, 1z, cin;   /* имената на променливите не могат да започват с цифра */
y = 1z;
x+y = 25.5;  // левият аргумент трябва да бъде променлива
cin = 5.5;
cin >> x; // cin е променлива, std::cin е оператор за вход
5.5 = x;  // левият аргумент трябва да бъде променлива
cout >> x >> y >> cin; /* посоката на стрелките е обърната
                          cout << x << y << cin;
                          */
```

Задача 3. Намерете и обяснете грешките в програмата на C++:

```
#include <iostream>  // липсва #
Using namespace std; // малка буква

const double FI = 1.5 + x; // x не е дефинирано

int main() {
  double a, b, c;
  cin << a, b, c;  // cin >> a >> b >> c;
  FI = a+b+c;      // опит за промяна на константа
  cout << FI << "\n";
                   // липсва return 0;
}
```

Манипулатори

- #include <iomanip>
- setw(<цял израз>) - задава ширината на полето на следващия вход/изход.

Забележка. Отнася се само за първата входна /изходна операция

Пример:

```
int a = 1234, b = 5;
cout << setw(10) << a << b;
```

```
изход
      12345
```

Манипулатори

Пример:

```
int a = 1234, b = 5;
cout << setw(10) << a << setw(5) << b;
```

изход

```
______1234____5
```

Манипулатори

- left, right, internal – подравняване на изхода.
  Забележка. Използва се само за изход

Пример:

```
int number = -1;
cout << setw(10) << number << endl;
cout << left << setw(10) << number << endl;
cout << right << setw(10) << number << endl;
cout << internal << setw(10) << number << endl;

// изход
_________-1
-1_________
_________-1
-_________1
```

Манипулатори

- setfill(<char>) - задава символ за запълване на полето.
  Забележка. Използва се само за изход

Пример:

```
int number = 1;
cout << setfill('*') << setw(10) << number
<< endl;

// изход
*********1
```

Формат на реалните числа

- Основен (общ) формат

12.456

- Експоненциален (научен, scientific) формат

1.2456e+001

- Фиксиран (с фиксирана точка, fixed)

Формат на реалните числа

Точност

Пример:

38.59417862 с точност 7

- в основен формат – броят на цифрите в цялата и дробната част е 7

  38.59418

- в експоненциален формат (scientific) – броят на цифрите след десетичната точка на мантисата е 7

  3.8594179e+001

- във фиксиран формат (fixed) – броят на цифрите след десетичната точка е 7
  38.5941786

Манипулатори

- scientific/fixed

```
double x = 38.59417862;
cout << scientific << x;
cout << fixed << x;
```

- смяна на основния формат

setiosflags(формат)

Пример:

```
cout << setiosflags(ios::scientific);
cout << setiosflags(ios::fixed);
```

минаване в основен формат

```
cout << resetiosflags(ios::scientific);
cout << resetiosflags(ios::fixed);
```

Манипулатори

- setprecision(int) – задава точността в десетична бройна система

Забележка:

Задава точността до указване на друга точност.

Възможно е закръгляне на числата.

Точността по подразбиране е 6.

Пример:

```
double x = 38.59417862;
cout << setprecision(7);
cout << x << endl;                  // 38.59418
cout << scientific << x;            // 3.8594179e+001
cout << fixed << x << endl;         // 38.5941786
```

Проблеми

Стойност извън размера на отделената памет

Не се съобщава за грешка

- Получава се стойност, която е неочаквана
- Резултатът се отрязва

Пример:

```
int x = 2000000000;
cout << 2*x; // -294967296
```

Масиви

- Крайна редица от фиксиран брой елементи от един и същ тип

```
T <променлива>[size] =
{<редица_от_константни_изрази>}опц;
```

Пример:

```
int test[5] = {1, 2, 3, 4, 5};
```

Масиви

Пример:

```
int test[5] = {1, 2, 3, 4, 5};
```

```
test == addr1
```

Масив

Пример:

```
int test[5] = {1, 2, 3, 4, 5};
```

```
test == addr1
```


```
test
```

Многомерни масиви

- Масив, базовият тип на който е едномерен масив, се нарича ***двумерен***.
- Масив, базовият тип на който е двумерен масив, се нарича ***тримерен***.
- Аналогично могат да се дефинират n-мерни масиви.

Многомерни масиви

```
T <променлива>[size1][size2] … [sizen]
```

- T – име или дефиниция на произволен тип

(без псевдоним, void и функционален)

- променлива – идентификатор
- size1, size2 … sizen – константни изрази от интегрален или изброен тип с положителни стойности

Пример:

```
int name[5][3] /*двумерен масив с елементи
                 от тип int */
```

Многомерни масиви

Пример:

```
int name[5][3];
```

`name[i]` - масив с 3 елем. от тип int, $i \in [0,4]$

`name[i][j]`
$i \in [0,4], j \in [0,2]$
стойност на позиция i, j

Многомерни масиви

Пример:

```
T name[size1][size2] = {
   {T0,0, T0,1, …, T0,size2-1},
   {T1,0, T1,1, …, T1,size2-1}
   …
   {Tsize1-1,0, Tsize1-1,1, …, Tsize1-1,size2-1}
}
```

Задача:

Да се дефинира матрица от цели числа с размерност 5x4.

Да се въведат стойности за елементите му от клавиатурата.

Да се изведат въведените стойности.

```
int matrix[5][4];

for(int i=0; i < 5; i++)
  for(int j=0; j < 4; j++)
    cin >> matrix[i][j];

for(int i=0; i < 5; i++) {
  for(int j=0; j < 4; j++)
    cout << matrix[i][j] << " ";
  cout << endl;
}
```

Задача:

Да се напише програмен фрагмент, който обхожда квадратна матрица по диагонали, започвайки от a_{00}

```
const int n = 4;
int matrix[n][n] = {
  { 1,  2,  3,  4},
  { 5,  6,  7,  8},
  { 9, 10, 11, 12},
  {13, 14, 15, 16}
};

for(int k=0; k < n; k++) {
  for(int i=k; i >= 0; i--)
    cout << matrix[i][k-i] << " ";
  cout << endl;
}

for(int k=n; k < 2*n-2; k++) {
  for(int i=n-1; i >= k-n+1; i--)
    cout << matrix[i][k-i] << " ";
  cout << endl;
}
```

Указатели

Оператор &

&<променлива>

- променлива – вече дефинирана променлива

Семантика

Намира адреса на променливата

Указатели

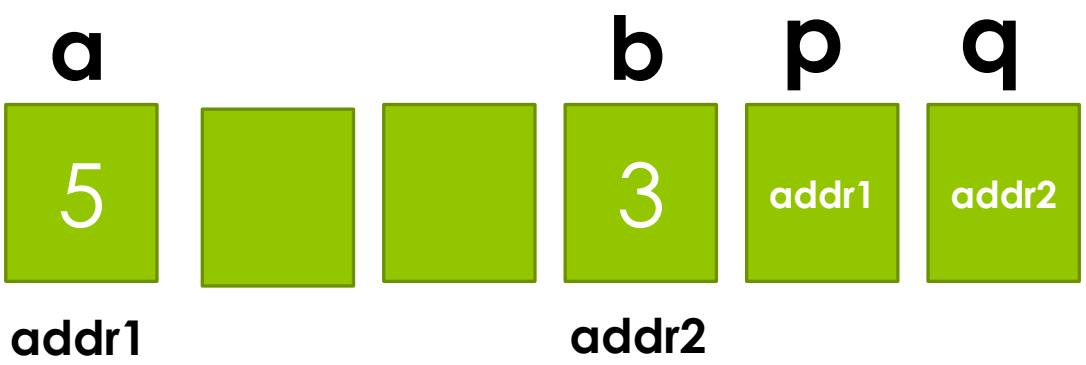
T* <променлива> [= <стойност>]$_{опц}$

T* – указател към променлива от тип T

- T – име или дефиниция на тип
- променлива – идентификатор
- стойност – шестнадесетично число, представляващо адрес на данна от тип T или NULL

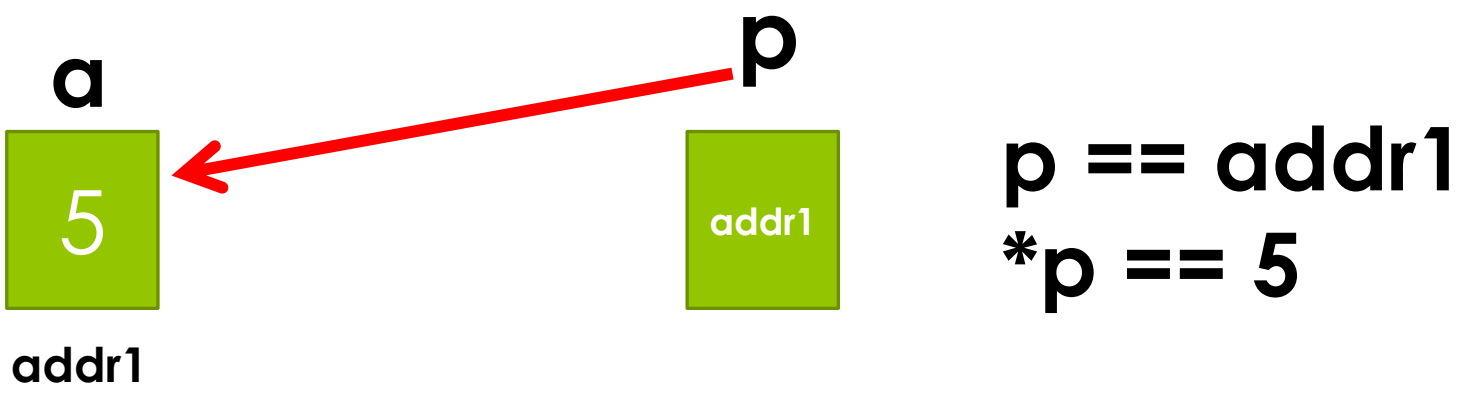
Указатели

```
int a = 5;
int b = 3
int *p = &a;
int *q = &q;
```



Указатели

```
int a = 5;
int *p = &a;
```



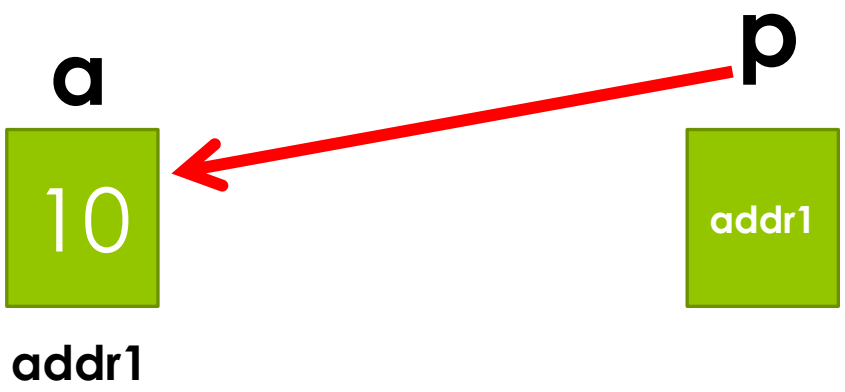
p == addr1

***p == 5**

Указатели

```
int a = 5;
int *p = &a;

*p = 10; // присвояване на нова стойност за a
```



Указатели

Забележка:

Дефиницията:
T a, b;*
е еквивалентна на:
T a;*
T b;

Дефиницията:
*T *a, *b;*
е еквивалентна на:
T a;*
T b;*

Аритметични и логически операции

- `+, -, ++, --, ==, !=, >, >=, <, <=`

Забележка

Не е възможно въвеждане на данни от тип указател чрез оператора >>

Адресна аритметика

```
int *p;
double *q;
```

```
p = p + 1;
```

премества указателя

p = p + 1*4,

4 е броят на байтовете, необходими за записване на данна от тип int

```
q = q + 1;
```

премества указателя

q = q + 1*8,

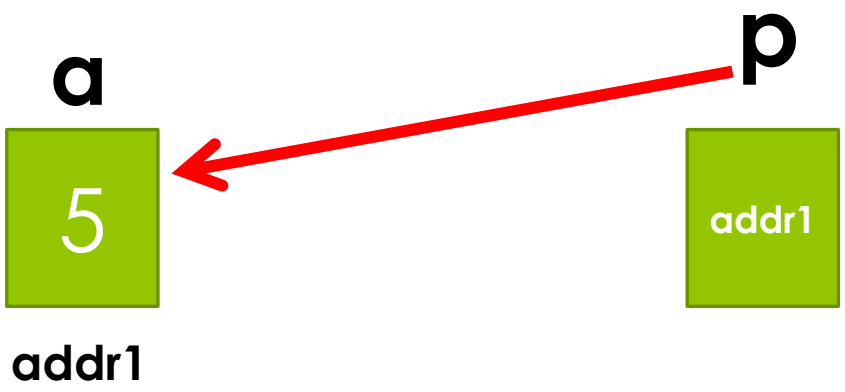
8 е броят на байтовете, необходими за записване на данна от тип double

Указатели

- Изравняване на типове

```
int *p = &a;
```

a – int
p – int*
***p – int**



Задача:

Да се изведат стойностите на едномерен и двумерен масив.

Да се използват указатели.

```
int matrix[5][4] = {
  { 1,  2,  3,  4},
  { 5,  6,  7,  8},
  { 9, 10, 11, 12},
  {13, 14, 15, 16},
  {23, 24, 25, 26}
};

for(int i=0; i < 5; i++) {
  for(int j=0; j < 4; j++)
    cout << *(*(matrix + i)+j) << " ";
  cout << endl;
}
```

```
cout << “Край”;
```