Указатели и препратки

Трифон Трифонов

Увод в програмирането,
спец. Компютърни науки, 1 поток, 2018/19 г.

13–20 декември 2018 г.

Тип указател

- **Множество от стойности:** всички възможни lvalue от даден тип и специалната стойност `nullptr`.

Тип указател

- **Множество от стойности:** всички възможни lvalue от даден тип и специалната стойност `nullptr`.
- Интегрален нечислов тип

Тип указател

- **Множество от стойности:** всички възможни lvalue от даден тип и специалната стойност `nullptr`.
- Интегрален нечислов тип
- Параметризиран тип: ако `T` е тип данни, то `T*` е тип "указател към елемент от тип `T`"

Тип указател

- **Множество от стойности:** всички възможни lvalue от даден тип и специалната стойност `nullptr`.
- Интегрален нечислов тип
- Параметризиран тип: ако `T` е тип данни, то `T*` е тип "указател към елемент от тип `T`"
- Физическо представяне: цяло число, указващо адреса на указваната lvalue в паметта

Тип указател

- **Множество от стойности:** всички възможни lvalue от даден тип и специалната стойност `nullptr`.
- Интегрален нечислов тип
- Параметризиран тип: ако `T` е тип данни, то `T*` е тип "указател към елемент от тип `T`"
- Физическо представяне: цяло число, указващо адреса на указваната lvalue в паметта
- Стойностите от тип "указател" са с размера на машинната дума

Тип указател

- **Множество от стойности:** всички възможни lvalue от даден тип и специалната стойност `nullptr`.
- Интегрален нечислов тип
- Параметризиран тип: ако `T` е тип данни, то `T*` е тип "указател към елемент от тип `T`"
- Физическо представяне: цяло число, указващо адреса на указваната lvalue в паметта
- Стойностите от тип "указател" са с размера на машинната дума
  - 32 бита (4 байта) за 32-битови процесорни архитектури

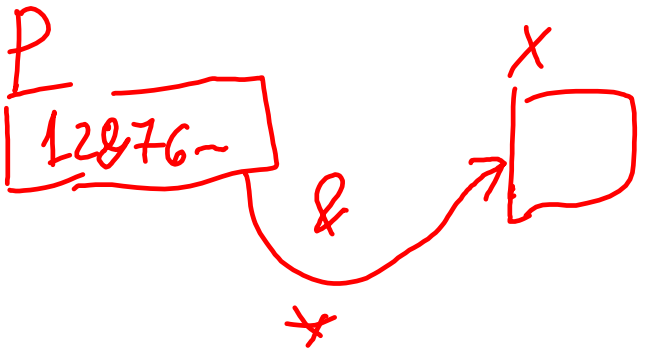
Тип указател

- **Множество от стойности:** всички възможни lvalue от даден тип и специалната стойност `nullptr`.
- Интегрален нечислов тип
- Параметризиран тип: ако `T` е тип данни, то `T*` е тип "указател към елемент от тип `T`"
- Физическо представяне: цяло число, указващо адреса на указваната lvalue в паметта
- Стойностите от тип "указател" са с размера на машинната дума
  - 32 бита (4 байта) за 32-битови процесорни архитектури
  - 64 бита (8 байта) за 64-битови процесорни архитектури

Операции с указатели

- реферирane (&<lvalue>)

Операции с указатели



- реферииране (&<lvalue>)
- дереферииране (*<указател>)
  - унарна операция!

Операции с указатели

- реферииране (&<lvalue>)
- дереферииране (*<указател>)
  - унарна операция!
- сравнение (==, !=, <, >, <=, >=)

Операции с указатели

- реферииране (&<lvalue>)
- дереферииране (*<указател>)
  - унарна операция!
- сравнение (==, !=, <, >, <=, >=)
- указателна аритметика (+,-,+=,-=,++,--)

Операции с указатели

- реферииране (&<lvalue>)
- дереферииране (*<указател>)
  - унарна операция!
- сравнение (==, !=, <, >, <=, >=)
- указателна аритметика (+,-,+=,-=,++,--)
- извеждане (<<)

Операции с указатели

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

Дефиниране на указателни променливи

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

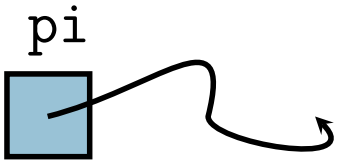
Примери:

Дефиниране на указателни променливи

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

Примери:

- `int *pi;`

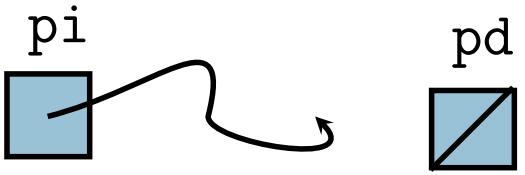


Дефиниране на указателни променливи

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

Примери:

- `int *pi;`
- `double *pd = nullptr;`

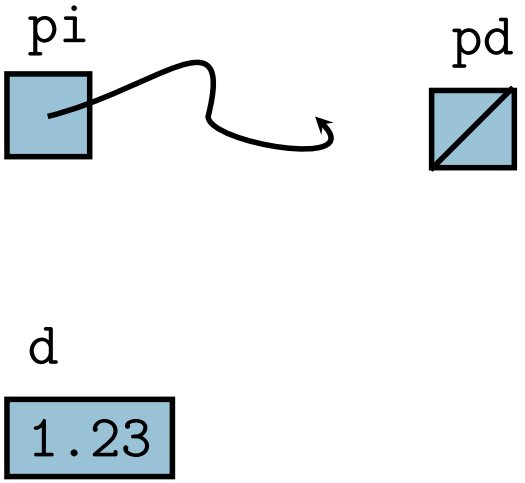


Дефиниране на указателни променливи

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

Примери:

- `int *pi;`
- `double *pd = nullptr;`
- `double d = 1.23;`

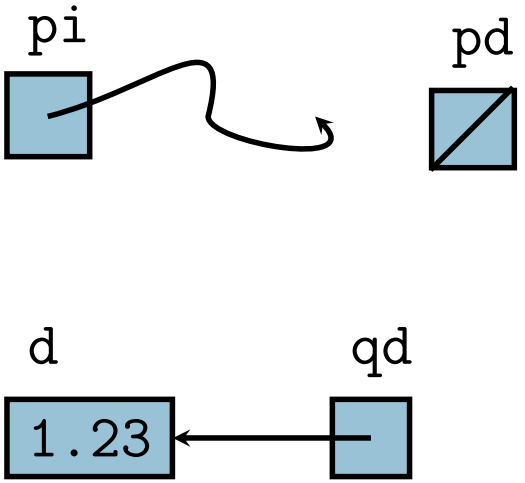


Дефиниране на указателни променливи

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

Примери:

- `int *pi;`
- `double *pd = nullptr;`
- `double d = 1.23;`
- `double *qd = &d;`

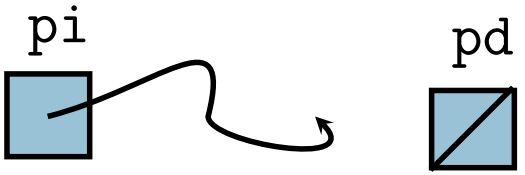


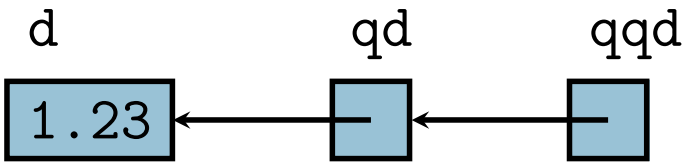
Дефиниране на указателни променливи

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

Примери:

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





Реферирane и дереферирane

- &<име> — указател към променливата <име>

Реферирane и дереферирane

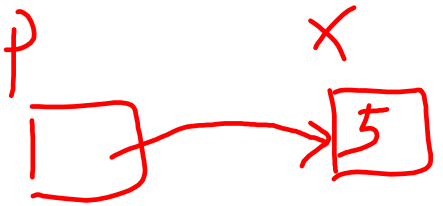
- &<име> — указател към променливата <име>
- *<указател> — мястото в паметта, сочено от <указател>

Реферирање и дереферирање

- &<име> — указател към променливата <име>
- *<указател> — мястото в паметта, сочено от <указател>
- **Примери:**

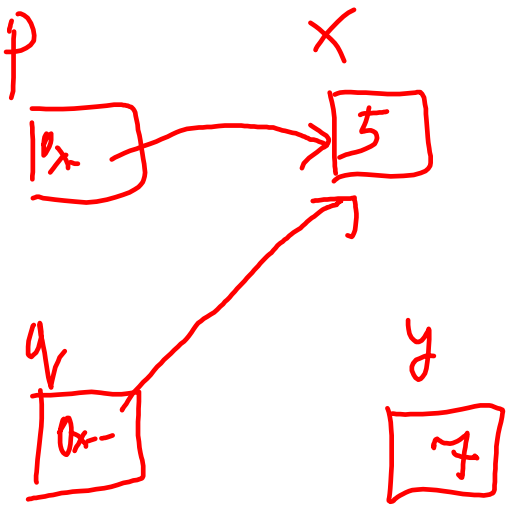
Реферииране и дереферииране

- &<име> — указател към променливата <име>
- *<указател> — мястото в паметта, сочено от <указател>
- **Примери:**
  - `int x = 5, *p = &x;`



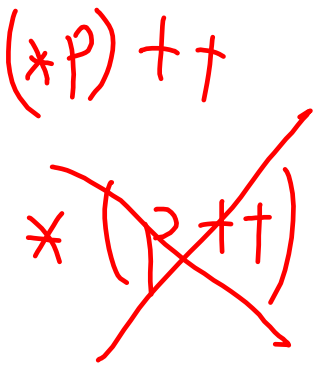
Рефериране и дерефериране

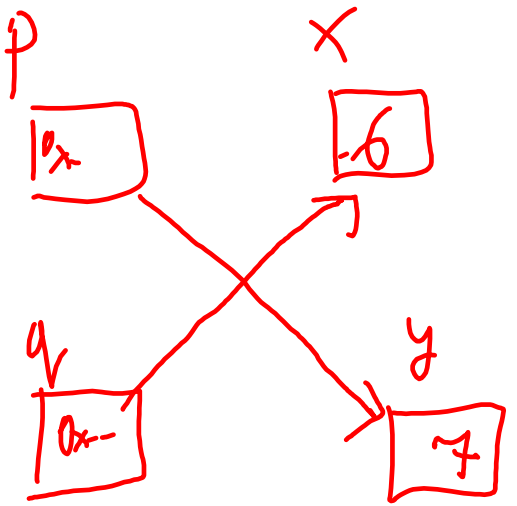
- &<име> — указател към променливата <име>
- *<указател> — мястото в паметта, сочено от <указател>
- **Примери:**
  - `int x = 5, *p = &x;`
  - `int *q = p, y = *p + 2;`



Рефериране и дерефериране

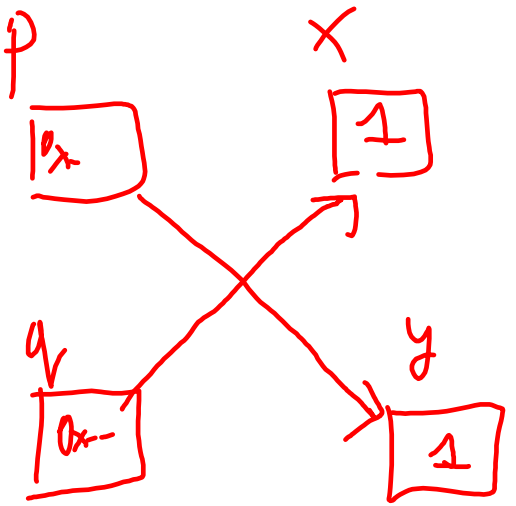
- &<име> — указател към променливата <име>
- *<указател> — мястото в паметта, сочено от <указател>
- **Примери:**
  - `int x = 5, *p = &x;`
  - `int *q = p, y = *p + 2;`
  - `*p++; p = &y;`





Рефериране и дерефериране

- &<име> — указател към променливата <име>
- *<указател> — мястото в паметта, сочено от <указател>
- **Примери:**
  - `int x = 5, *p = &x;`
  - `int *q = p, y = *p + 2;`
  - `*p++; p = &y;`
  - `*q = 1; *p = *q;`



Реферираване и дерефериране

- &<име> — указател към променливата <име>
- *<указател> — мястото в паметта, сочено от <указател>
- **Примери:**
  - `int x = 5, *p = &x;`
  - `int *q = p, y = *p + 2;`
  - `*p++; p = &y;`
  - `*q = 1; *p = *q;`
- &<lvalue> връща като резултат <rvalue>!

Реферирane и дереферирane

- &<име> — указател към променливата <име>
- *<указател> — мястото в паметта, сочено от <указател>
- **Примери:**
  - `int x = 5, *p = &x;`
  - `int *q = p, y = *p + 2;`
  - `*p++; p = &y;`
  - `*q = 1; *p = *q;`
- &<lvalue> връща като резултат <rvalue>!
  - ~~&3~~

Рефериране и дерефериране

- &<име> — указател към променливата <име>
- *<указател> — мястото в паметта, сочено от <указател>
- **Примери:**
  - `int x = 5, *p = &x;`
  - `int *q = p, y = *p + 2;`
  - `*p++; p = &y;`
  - `*q = 1; *p = *q;`
- &<lvalue> връща като резултат <rvalue>!
  - ~~`&3`~~
  - ~~`&x = p;`~~

Рефериране и дерефериране

- &<име> — указател към променливата <име>
- *<указател> — мястото в паметта, сочено от <указател>
- **Примери:**
  - `int x = 5, *p = &x;`
  - `int *q = p, y = *p + 2;`
  - `*p++; p = &y;`
  - `*q = 1; *p = *q;`
- &<lvalue> връща като резултат <rvalue>!
  - ~~`&3`~~
  - ~~`&x = p;`~~
- *<rvalue> връща като резултат <lvalue>!

Реферирane и дереферирane

- &<име> — указател към променливата <име>
- *<указател> — мястото в паметта, сочено от <указател>
- **Примери:**
  - `int x = 5, *p = &x;`
  - `int *q = p, y = *p + 2;`
  - `*p++; p = &y;`
  - `*q = 1; *p = *q;`
- &<lvalue> връща като резултат <rvalue>!
  - ~~`&3`~~
  - ~~`&x = p;`~~
- *<rvalue> връща като резултат <lvalue>!
  - `*p = x;`

Рефериране и дерефериране

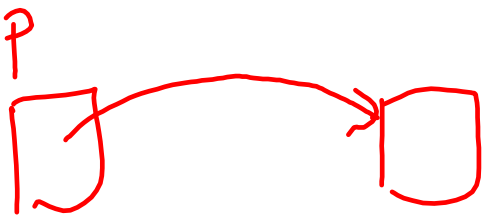
- &<име> — указател към променливата <име>
- *<указател> — мястото в паметта, сочено от <указател>
- **Примери:**
  - `int x = 5, *p = &x;`
  - `int *q = p, y = *p + 2;`
  - `*p++; p = &y;`
  - `*q = 1; *p = *q;`
- &<lvalue> връща като резултат <rvalue>!
  - ~~`&3`~~
  - ~~`&x = p;`~~
- *<rvalue> връща като резултат <lvalue>!
  - `*p = x;`
  - `**qqd = 3.15;`

Рефериране и дерефериране

- &<име> — указател към променливата <име>
- *<указател> — мястото в паметта, сочено от <указател>
- **Примери:**
  - `int x = 5, *p = &x;`
  - `int *q = p, y = *p + 2;`
  - `*p++; p = &y;`
  - `*q = 1; *p = *q;`
- &<lvalue> връща като резултат <rvalue>!
  - ~~`&3`~~
  - ~~`&x = p;`~~
- *<rvalue> връща като резултат <lvalue>!
  - `*p = x;`
  - `**qqd = 3.15;`
- операциите са дуални една на друга и се унищожават взаимно

Реферииране и дереферииране

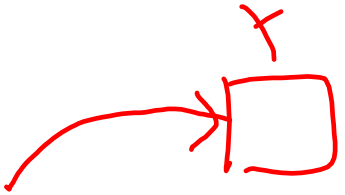
- &<име> — указател към променливата <име>
- *<указател> — мястото в паметта, сочено от <указател>
- **Примери:**
  - `int x = 5, *p = &x;`
  - `int *q = p, y = *p + 2;`
  - `*p++; p = &y;`
  - `*q = 1; *p = *q;`



- &<lvalue> връща като резултат <rvalue>!
  - ~~`&3`~~
  - ~~`&x = p;`~~
- *<rvalue> връща като резултат <lvalue>!
  - `*p = x;`
  - `**qqd = 3.15;`
- операциите са дуални една на друга и се унищожават взаимно
  - `&(*p)` $\Longleftrightarrow$ `p`

Реферииране и дереферииране

- &<име> — указател към променливата <име>
- *<указател> — мястото в паметта, сочено от <указател>
- **Примери:**
  - `int x = 5, *p = &x;`
  - `int *q = p, y = *p + 2;`
  - `*p++; p = &y;`
  - `*q = 1; *p = *q;`



- &<lvalue> връща като резултат <rvalue>!
  - ~~`&3`~~
  - ~~`&x = p;`~~
- *<rvalue> връща като резултат <lvalue>!
  - `*p = x;`
  - `**qqd = 3.15;`
- операциите са дуални една на друга и се унищожават взаимно
  - `&(*p)` $\Longleftrightarrow$ `p`
  - `*(&x)` $\Longleftrightarrow$ `x`