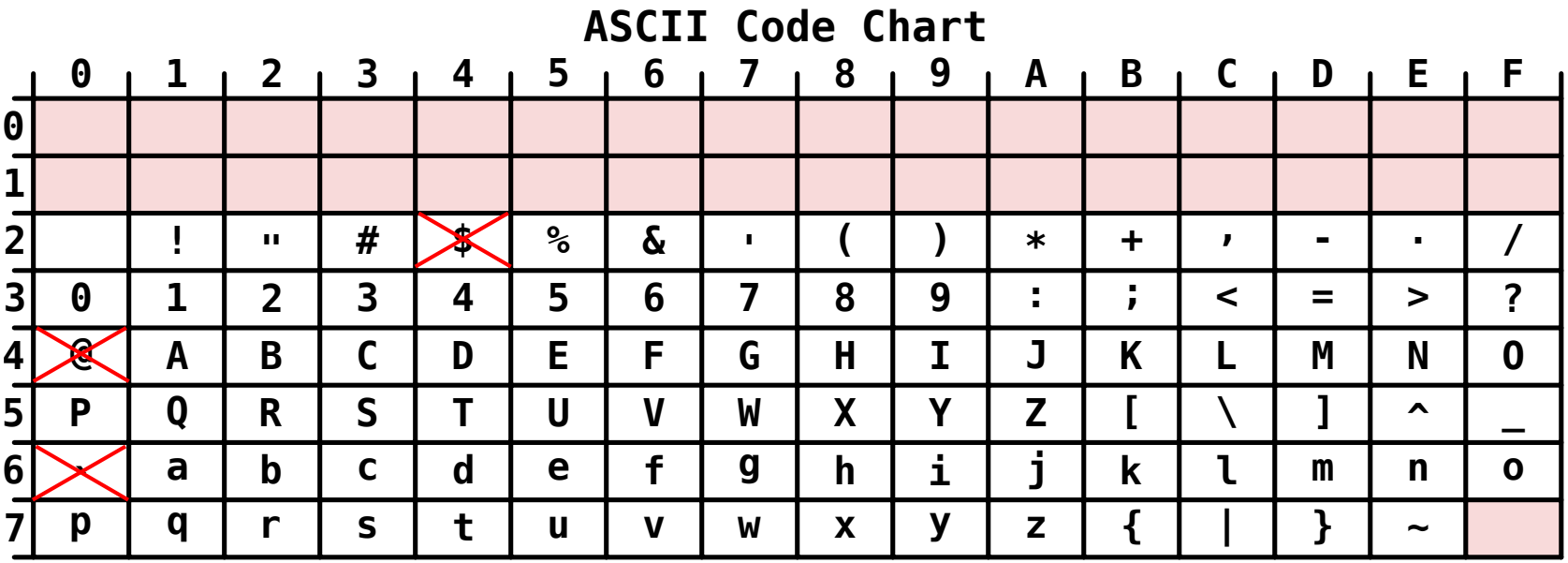
Основни елементи на C++

Трифон Трифонов

Увод в програмирането,
спец. Компютърни науки, 1 поток,
спец. Софтуерно инженерство,
2016/17 г.

12–19 октомври 2016 г.

Азбука



Изображения: wikipedia.org

Синтаксис

- Правила за построяване на текст

Синтаксис

- Правила за построяване на текст
- Иван чете интересна книга.

Синтаксис

- Правила за построяване на текст
- Иван чете интересна книга.
- Студентът пише програма.

Синтаксис

- Правила за построяване на текст
- Иван чете интересна книга.
- Студентът пише програма.
- книга. чете Иван? интерес на

Синтаксис

- Правила за построяване на текст
- Иван чете интересна книга.
- Студентът пише програма.
- книга. чете Иван? интерес на
- <изречение> ::= <подлог> <сказуемо> [<определение>] <допълнение>.

Синтаксис

- Правила за построяване на текст
- Иван чете интересна книга.
- Студентът пише програма.
- книга. чете Иван? интерес на
- <изречение> ::= <подлог> <сказуемо> [<определение>] <допълнение>.
- <подлог> ::= <собствено_съществително> | <нарицателно_съществително><пълен_член>

Синтаксис

- Правила за построяване на текст
- Иван чете интересна книга.
- Студентът пише програма.
- книга. чете Иван? интерес на
- <изречение> ::= <подлог> <сказуемо> [<определение>] <допълнение>.
- <подлог> ::= <собствено_съществително> | <нарицателно_съществително><пълен_член>
- <пълен_член> ::= ът | ят | та | то

Синтаксис

- Правила за построяване на текст
- Иван чете интересна книга.
- Студентът пише програма.
- книга. чете Иван? интерес на
- <изречение> ::= <подлог> <сказуемо> [<определение>] <допълнение>.
- <подлог> ::= <собствено_съществително> | <нарицателно_съществително><пълен_член>
- <пълен_член> ::= ът | ят | та | то
- <сказуемо> ::= <глагол>

Синтаксис

- Правила за построяване на текст
- Иван чете интересна книга.
- Студентът пише програма.
- книга. чете Иван? интерес на
- <изречение> ::= <подлог> <сказуемо> [<определение>] <допълнение>.
- <подлог> ::= <собствено_съществително> | <нарицателно_съществително><пълен_член>
- <пълен_член> ::= ът | ят | та | то
- <сказуемо> ::= <глагол>
- <определение> ::= <прилагателно>

Синтаксис

- Правила за построяване на текст
- Иван чете интересна книга.
- Студентът пише програма.
- книга. чете Иван? интерес на
- <изречение> ::= <подлог> <сказуемо> [<определение>] <допълнение>.
- <подлог> ::= <собствено_съществително> | <нарицателно_съществително><пълен_член>
- <пълен_член> ::= ът | ят | та | то
- <сказуемо> ::= <глагол>
- <определение> ::= <прилагателно>
- <допълнение> ::= <собствено_съществително> | <нарицателно_съществително>

Синтактичен анализ — пример 1

- <изречение>

Синтактичен анализ — пример 1

- <изречение>
- <подлог> <сказуемо> [<определение>] <допълнение>.

Синтактичен анализ — пример 1

- <изречение>
- <подлог> <сказуемо> [<определение>] <допълнение>.
- <собствено_съществително> <сказуемо> <определение> <допълнение>.

Синтактичен анализ — пример 1

- <изречение>
- <подлог> <сказуемо> [<определение>] <допълнение>.
- <собствено_съществително> <сказуемо> <определение> <допълнение>.
- Иван <глагол> <определение> <допълнение>.

Синтактичен анализ — пример 1

- <изречение>
- <подлог> <сказуемо> [<определение>] <допълнение>.
- <собствено_съществително> <сказуемо> <определение> <допълнение>.
- Иван <глагол> <определение> <допълнение>.
- Иван чете <определение> <нарицателно_съществително>.

Синтактичен анализ — пример 1

- <изречение>
- <подлог> <сказуемо> [<определение>] <допълнение>.
- <собствено_съществително> <сказуемо> <определение> <допълнение>.
- Иван <глагол> <определение> <допълнение>.
- Иван чете <определение> <нарицателно_съществително>.
- Иван чете <прилагателно> книга.

Синтактичен анализ — пример 1

- <изречение>
- <подлог> <сказуемо> [<определение>] <допълнение>.
- <собствено_съществително> <сказуемо> <определение> <допълнение>.
- Иван <глагол> <определение> <допълнение>.
- Иван чете <определение> <нарицателно_съществително>.
- Иван чете <прилагателно> книга.
- Иван чете интересна книга.

Синтактичен анализ — пример 2

- <изречение>

Синтактичен анализ — пример 2

- <изречение>
- <подлог> <сказуемо> [<определение>] <допълнение>.

Синтактичен анализ — пример 2

- <изречение>
- <подлог> <сказуемо> [<определение>] <допълнение>.
- <нарицателно_съществително><пълен_член> <сказуемо> <допълнение>.

Синтактичен анализ — пример 2

- <изречение>
- <подлог> <сказуемо> [<определение>] <допълнение>.
- <нарицателно_съществително><пълен_член> <сказуемо> <допълнение>.
- Студент<пълен_член> <глагол> <допълнение>.

Синтактичен анализ — пример 2

- <изречение>
- <подлог> <сказуемо> [<определение>] <допълнение>.
- <нарицателно_съществително><пълен_член> <сказуемо> <допълнение>.
- Студент<пълен_член> <глагол> <допълнение>.
- Студентът <глагол> <нарицателно_съществително>.

Синтактичен анализ — пример 2

- <изречение>
- <подлог> <сказуемо> [<определение>] <допълнение>.
- <нарицателно_съществително><пълен_член> <сказуемо> <допълнение>.
- Студент<пълен_член> <глагол> <допълнение>.
- Студентът <глагол> <нарицателно_съществително>.
- Студентът <глагол> програма.

Синтактичен анализ — пример 2

- <изречение>
- <подлог> <сказуемо> [<определение>] <допълнение>.
- <нарицателно_съществително><пълен_член> <сказуемо> <допълнение>.
- Студент<пълен_член> <глагол> <допълнение>.
- Студентът <глагол> <нарицателно_съществително>.
- Студентът <глагол> програма.
- Студентът пише програма.

Синтактичен анализ — пример 3

- <изречение>

Синтактичен анализ — пример 3

- <изречение>
- <подлог> <сказуемо> [<определение>] <допълнение>.

Синтактичен анализ — пример 3

- <изречение>
- <подлог> <сказуемо> [<определение>] <допълнение>.
- <нарицателно_съществително><пълен_член> <сказуемо> <собствено_съществително>.

Синтактичен анализ — пример 3

- <изречение>
- <подлог> <сказуемо> [<определение>] <допълнение>.
- <нарицателно_съществително><пълен_член> <сказуемо> <собствено_съществително>.
- Програма<пълен_член> <глагол> Иван.

Синтактичен анализ — пример 3

- <изречение>
- <подлог> <сказуемо> [<определение>] <допълнение>.
- <нарицателно_съществително><пълен_член> <сказуемо> <собствено_съществително>.
- Програма<пълен_член> <глагол> Иван.
- Програмата гледа Иван.

Синтактичен анализ — пример 3

- <изречение>
- <подлог> <сказуемо> [<определение>] <допълнение>.
- <нарицателно_съществително><пълен_член> <сказуемо> <собствено_съществително>.
- Програма<пълен_член> <глагол> Иван.
- Програмата гледа Иван.

Синтактичен анализ — пример 3

- <изречение>
- <подлог> <сказуемо> [<определение>] <допълнение>.
- <нарицателно_съществително><пълен_член> <сказуемо> <собствено_съществително>.
- Програма<пълен_член> <глагол> Иван.
- Програмата гледа Иван.

- Освен да е построено правилно, изречението трябва да има смисъл!

Синтактичен анализ — пример 3

- <изречение>
- <подлог> <сказуемо> [<определение>] <допълнение>.
- <нарицателно_съществително><пълен_член> <сказуемо> <собствено_съществително>.
- Програма<пълен_член> <глагол> Иван.
- Програмата гледа Иван.

- Освен да е построено правилно, изречението трябва да има смисъл!
- **Семантика:** смисъл, значение на текст

Мета-език на Backus-Naur

- <цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Мета-език на Backus-Naur

- <цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
- <цяло_число_без_знак> ::= <цифра> {<цифра>}

Мета-език на Backus-Naur

- <цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
- <цяло_число_без_знак> ::= <цифра> {<цифра>}
- <цяло_число> ::= [+|-] <цяло_число_без_знак>

Мета-език на Backus-Naur

- <цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
- <цяло_число_без_знак> ::= <цифра> {<цифра>}
- <цяло_число> ::= [+|-] <цяло_число_без_знак>
  - -15, 2, +412

Мета-език на Backus-Naur

- <цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
- <цяло_число_без_знак> ::= <цифра> {<цифра>}
- <цяло_число> ::= [+|-] <цяло_число_без_знак>
  - -15, 2, +412
- <латинска_буква> ::= A | B | ... | Y | Z | a | b | ... | y | z

Мета-език на Backus-Naur

- <цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
- <цяло_число_без_знак> ::= <цифра> {<цифра>}
- <цяло_число> ::= [+|-] <цяло_число_без_знак>
  - -15, 2, +412
- <латинска_буква> ::= A | B | ... | Y | Z | a | b | ... | y | z
- <идентификатор> ::= _ | <латинска_буква> {<латинска_буква> | <цифра> | _ }

Мета-език на Backus-Naur

- <цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
- <цяло_число_без_знак> ::= <цифра> {<цифра>}
- <цяло_число> ::= [+|-] <цяло_число_без_знак>
  - -15, 2, +412
- <латинска_буква> ::= A | B | ... | Y | Z | a | b | ... | y | z
- <идентификатор> ::= _ | <латинска_буква> {<латинска_буква> | <цифра> | _ }
  - a, name, X1, _Data15

Основни думи на C++ (tokens)

- <идентификатор> ::= _ | <латинска_буква> {<латинска_буква> | <цифра> | _ }

Основни думи на C++ (tokens)

- <идентификатор> ::= _ | <латинска_буква> {<латинска_буква> | <цифра> | _ }
- запазени думи

Основни думи на C++ (tokens)

- <идентификатор> ::= _ | <латинска_буква> {<латинска_буква> | <цифра> | _ }
- запазени думи
- стандартни идентификатори

Основни думи на C++ (tokens)

- <идентификатор> ::= _ | <латинска_буква> {<латинска_буква> | <цифра> | _ }
- запазени думи
- стандартни идентификатори
- литерали

Основни думи на C++ (tokens)

- <идентификатор> ::= _ | <латинска_буква> {<латинска_буква> | <цифра> | _ }
- запазени думи
- стандартни идентификатори
- литерали
  - числови (`1`, `-5`, `+2.34`, `1e-02`, `012`, `0x123`)

Основни думи на C++ (tokens)

- <идентификатор> ::= _ | <латинска_буква> {<латинска_буква> | <цифра> | _ }
- запазени думи
- стандартни идентификатори
- литерали
  - числови (`1`, `-5`, `+2.34`, `1e-02`, `012`, `0x123`)
  - символни (`'a'`, `'\t'`)

Основни думи на C++ (tokens)

- <идентификатор> ::= _ | <латинска_буква> {<латинска_буква> | <цифра> | _ }
- запазени думи
- стандартни идентификатори
- литерали
  - числови (`1`, `-5`, `+2.34`, `1e-02`, `012`, `0x123`)
  - символни (`'a'`, `'\t'`)
  - низови (`"hello"`, `"yes!"`)

Основни думи на C++ (tokens)

- <идентификатор> ::= _ | <латинска_буква> {<латинска_буква> | <цифра> | _ }
- запазени думи
- стандартни идентификатори
- литерали
  - числови (`1`, `-5`, `+2.34`, `1e-02`, `012`, `0x123`)
  - символни (`'a'`, `'\t'`)
  - низови (`"hello"`, `"yes!"`)
- операции (`+`, `-`, `*`, `/`)

Основни думи на C++ (tokens)

- <идентификатор> ::= _ | <латинска_буква> {<латинска_буква> | <цифра> | _ }
- запазени думи
- стандартни идентификатори
- литерали
  - числови (`1`, `-5`, `+2.34`, `1e-02`, `012`, `0x123`)
  - символни (`'a'`, `'\t'`)
  - низови (`"hello"`, `"yes!"`)
- операции (`+`, `-`, `*`, `/`)
- разделители (`:` `;` `,` `(` `)` `[` `]` `{` `}` `<` `>`)

Коментари

- <коментар> ::= //<текст_без_нов_ред> | /* <текст> */

Коментари

- <коментар> ::= //<текст_без_нов_ред> | /* <текст> */
- Компилаторът игнорира:

Коментари

- <коментар> ::= //<текст_без_нов_ред> | /* <текст> */
- Компилаторът игнорира:
  - коментари

Коментари

- <коментар> ::= //<текст_без_нов_ред> | /* <текст> */
- Компилаторът игнорира:
  - коментари
  - празни символи (интервал, табулация, нов ред)

Коментари

- <коментар> ::= //<текст_без_нов_ред> | /* <текст> */
- Компилаторът игнорира:
  - коментари
  - празни символи (интервал, табулация, нов ред)
- Пример:

```
int sum = 0; // нулираме сумата
/*
   вече сме готови да започнем пресмятането
   последователно ще натрупваме поредните числа в sum
   докато не ги изчерпим всичките
*/
...
```

Променливи

Променливата е именувана област в паметта.

Различно от променлива в математиката!

Променливи

Променливата е именувана област в паметта.

Различно от променлива в математиката!

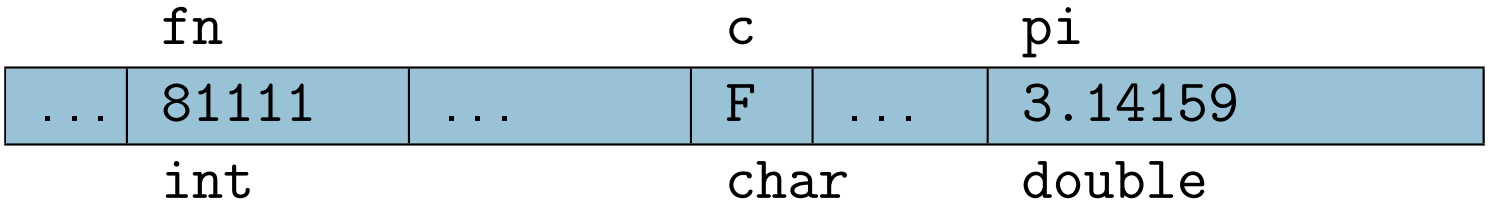
- Име (идентификатор)
- Място в паметта (адрес)
- Тип
- Стойност

Променливи

Променливата е именувана област в паметта.

Различно от променлива в математиката!

- Име (идентификатор)
- Място в паметта (адрес)
- Тип
- Стойност



Дефиниция и присвояване

<дефиниция> ::= <тип> <идентификатор> [= <израз>] {,
<идентификатор> [= <израз>] };

<присвояване> ::= <идентификатор> = <израз>;

Дефиниция и присвояване

<дефиниция> ::= <тип> <идентификатор> [= <израз>] {,
<идентификатор> [= <израз>] };

<присвояване> ::= <идентификатор> = <израз>;

Примери:

- `double x;`
- `int a, b = 15;`
- `a = b + 5;`
- `x = a * (b - 3);`
- ~~`double y = double x;`~~

Изход на екрана

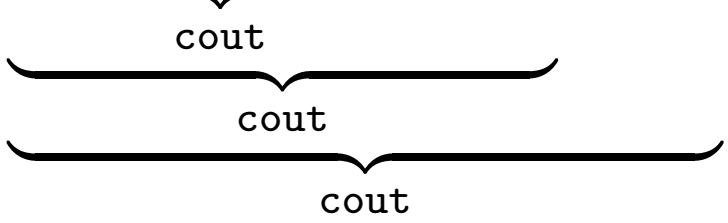
- `cout << <израз> {<< <израз>};`
- `cout << a << b << c;`

Изход на екрана

- `cout << <израз> {<< <израз>};`
- `((cout << a) << b) << c;`

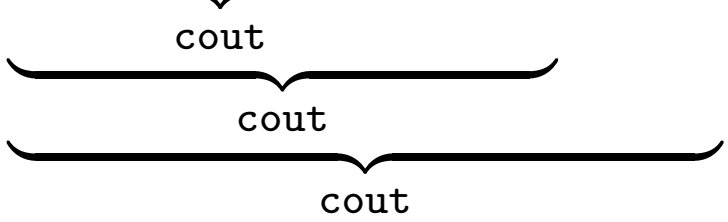
Изход на екрана

- `cout << <израз> {<< <израз>};`
- `( (cout << a) << b) << c ;`



Изход на екрана

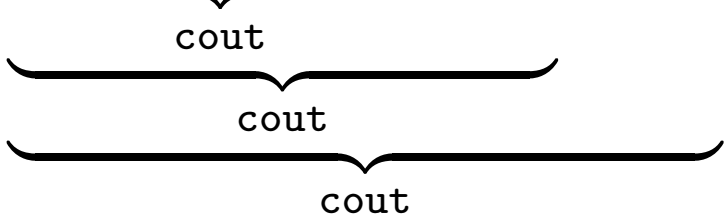
- `cout << <израз> {<< <израз>};`
- `( (cout << a) << b) << c ;`



- `cout << "a + b = " << a + b << endl;`

Изход на екрана

- `cout << <израз> {<< <израз>};`
- `( (cout << a) << b) << c ;`



- `cout << "a + b = " << a + b << endl;`
- ~~`cout << "a = " 2;`~~

Вход от клавиатурата

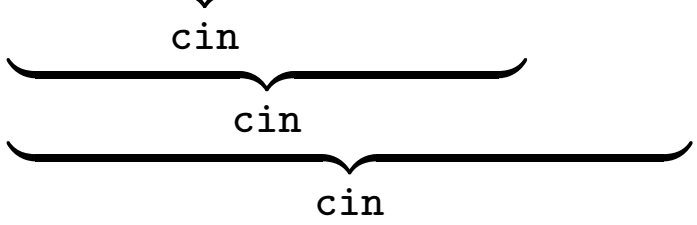
- `cin >>` <идентификатор> {`>>` <идентификатор>}`;`
- `cin >> a >> b >> c;`

Вход от клавиатурата

- `cin >>` <идентификатор> {`>>` <идентификатор>}`;`
- `((cin >> a) >> b) >> c;`

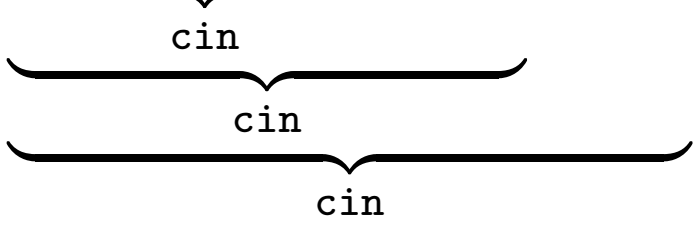
Вход от клавиатурата

- `cin >>` <идентификатор> {`>>` <идентификатор>};
- `( (cin >> a) >> b) >> c ;`



Вход от клавиатурата

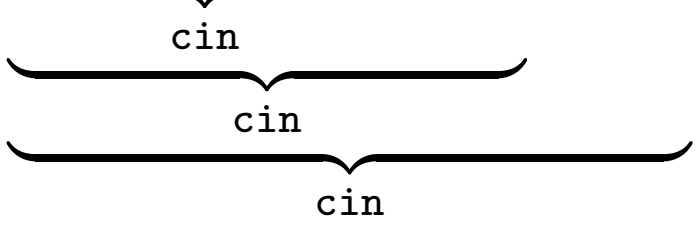
- `cin >>` <идентификатор> {`>>` <идентификатор>}`;`
- `( (cin >> a) >> b) >> c ;`



- ~~`cin >> a + b;`~~

Вход от клавиатурата

- `cin >>` <идентификатор> {`>>` <идентификатор>}`;`
- `( (cin >> a) >> b) >> c ;`



- ~~`cin >> a + b;`~~
- ~~`cin >> 15;`~~

Константи

- `const` <тип> <идентификатор> `=` <израз>`;`

Константи

- `const` <тип> <идентификатор> `=` <израз>`;`
- стойността на константите:

Константи

- `const` <тип> <идентификатор> `=` <израз>`;`
- стойността на константите:
  - трябва да бъде зададена при дефиниране

Константи

- `const` <тип> <идентификатор> `=` <израз>`;`
- стойността на константите:
  - трябва да бъде зададена при дефиниране
  - не може да се променя след това

Константи

- `const` <тип> <идентификатор> `=` <израз>`;`
- стойността на константите:
  - трябва да бъде зададена при дефиниране
  - не може да се променя след това
- Примери:

Константи

- `const` <тип> <идентификатор> `=` <израз>`;`
- стойността на константите:
  - трябва да бъде зададена при дефиниране
  - не може да се променя след това
- Примери:
  - `const int FINGERS = 10;`

Константи

- `const` <тип> <идентификатор> `=` <израз>`;`
- стойността на константите:
  - трябва да бъде зададена при дефиниране
  - не може да се променя след това
- Примери:
  - `const int FINGERS = 10;`
  - ~~`FINGERS = FINGERS + 2;`~~

Типове

- Класификация на видовете данни

Типове

- Класификация на видовете данни
- Носят **семантична** информация

Типове

- Класификация на видовете данни
- Носят **семантична** информация
- Помагат за проверка на коректност

Типове

- Класификация на видовете данни
- Носят **семантична** информация
- Помагат за проверка на коректност
- Множество от допустими стойности

Типове

- Класификация на видовете данни
- Носят **семантична** информация
- Помагат за проверка на коректност
- Множество от допустими стойности
- Операции

Типове

- Класификация на видовете данни
- Носят **семантична** информация
- Помагат за проверка на коректност
- Множество от допустими стойности
- Операции
- Вградени функции

Класификация на типовете

- Скаларни (атомарни)
  - интегрални
    - булев (`bool`)
    - целочислен (`int`)
    - символен (`char`)
    - изброен (`enum`)
  - други
    - числа с плаваща запетая (`float`, `double`)
    - указател (`T*`)
    - псевдоним (`T&`)
- Съставни
  - масив (`[]`)
    - низ (`char[]`)
  - структура (`struct`)
  - клас (`class`)
  - обединение (`union`)

Логически тип (bool)

- Множество от стойности: {false, true}
- <булева_константа> ::= true | false
- логически операции

Конюкнция

&&	false	true
false	false	false
true	false	true

Дизюкнция

||	false	true
false	false	true
true	true	true

Отрицание

!	
false	true
true	false

Символен тип (char)

- Множество от стойности
  - `signed char`: $[-128; 127]$
  - `unsigned char`: $[0; 255]$
- Литерали
  - '<символ>'
  - '\<контролен_символ>'

Целочислен тип (`int`)

- Множество от стойности: $[-2^{31}; 2^{31} - 1]$
- модификатори
  - `short`: $[-2^{15}; 2^{15} - 1]$
  - `long`: $[-2^{63}; 2^{63} - 1]$
  - `unsigned`: $[0; 2^x - 1]$, където $(x = 16, 32, 64)$

Целочислен тип (`int`)

- аритметични операции
  - едноместни операции за знак (+, -)
  - двуместни аритметични операции
    - `a + b` (събиране)
    - `a - b` (изваждане)
    - `a * b` (умножение)
    - `a / b` (частно)
    - `a % b` (остатък)
- операции за сравнение (предикати)
  - `a == b` (равно)
  - `a != b` (различно)
  - `a < b` (по-малко)
  - `a > b` (по-голямо)
  - `a <= b` (по-малко или равно)
  - `a >= b` (по-голямо или равно)

Числа с плаваща запетая

- Внимание: това не са реални числа!

Числа с плаваща запетая

- Внимание: това не са реални числа!
  - А какво са реални числа?

Числа с плаваща запетая

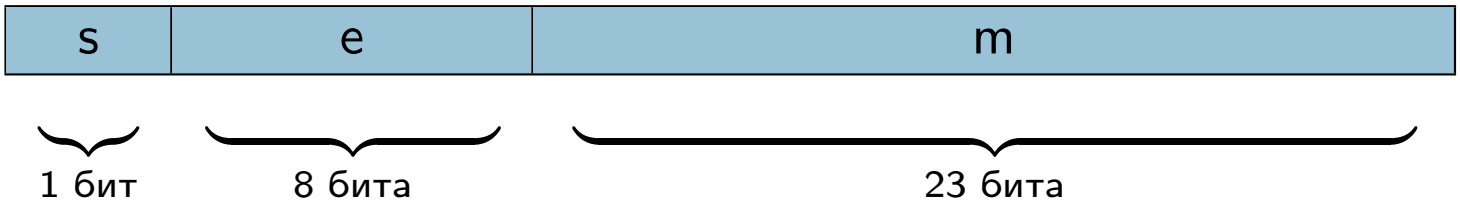
- Внимание: това не са реални числа!
  - А какво са реални числа?
- Още ще ги наричаме и **дробни числа**

Числа с плаваща запетая

- Внимание: това не са реални числа!
  - А какво са реални числа?
- Още ще ги наричаме и **дробни числа**
- Представяне в паметта

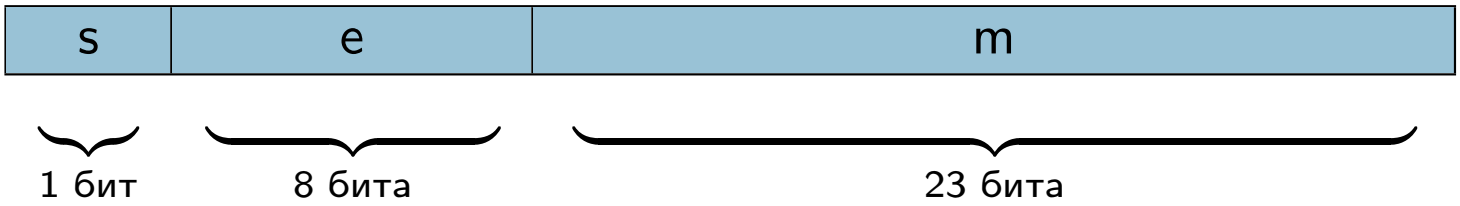
Числа с плаваща запетая

- Внимание: това не са реални числа!
  - А какво са реални числа?
- Още ще ги наричаме и **дробни числа**
- Представяне в паметта
  - $f = (-1)^s \cdot m \cdot 2^e$



Числа с плаваща запетая

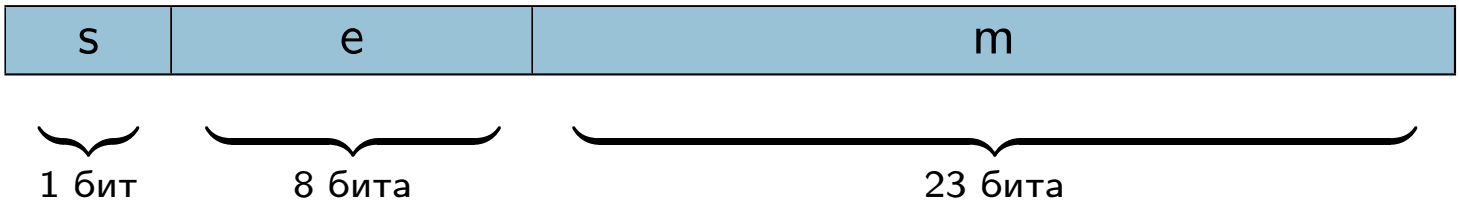
- Внимание: това не са реални числа!
  - А какво са реални числа?
- Още ще ги наричаме и **дробни числа**
- Представяне в паметта
  - $f = (-1)^s \cdot m \cdot 2^e$



  - $s \in \{0, 1\}$ — знак

Числа с плаваща запетая

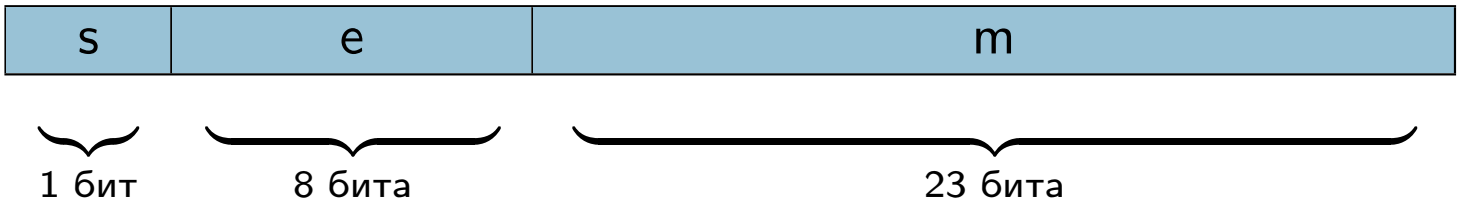
- Внимание: това не са реални числа!
  - А какво са реални числа?
- Още ще ги наричаме и **дробни числа**
- Представяне в паметта
  - $f = (-1)^s \cdot m \cdot 2^e$



  - $s \in \{0, 1\}$ — знак
  - $m \in [0; 2^{23} - 1]$ — мантиса

Числа с плаваща запетая

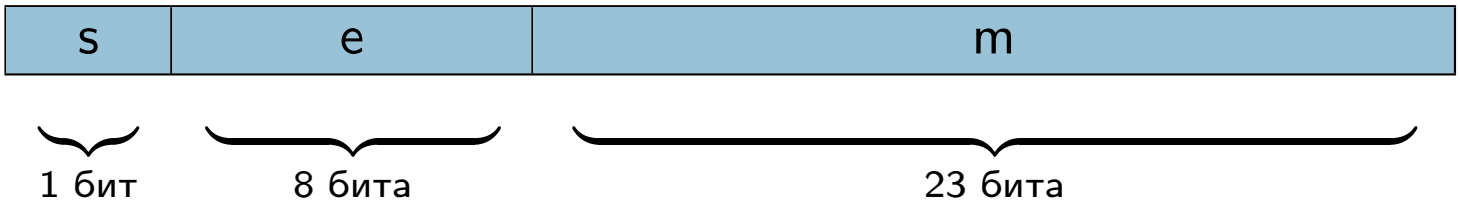
- Внимание: това не са реални числа!
  - А какво са реални числа?
- Още ще ги наричаме и **дробни числа**
- Представяне в паметта
  - $f = (-1)^s \cdot m \cdot 2^e$



  - $s \in \{0, 1\}$ — знак
  - $m \in [0; 2^{23} - 1]$ — мантиса
  - $e \in [-126; 127]$ — експонента

Числа с плаваща запетая

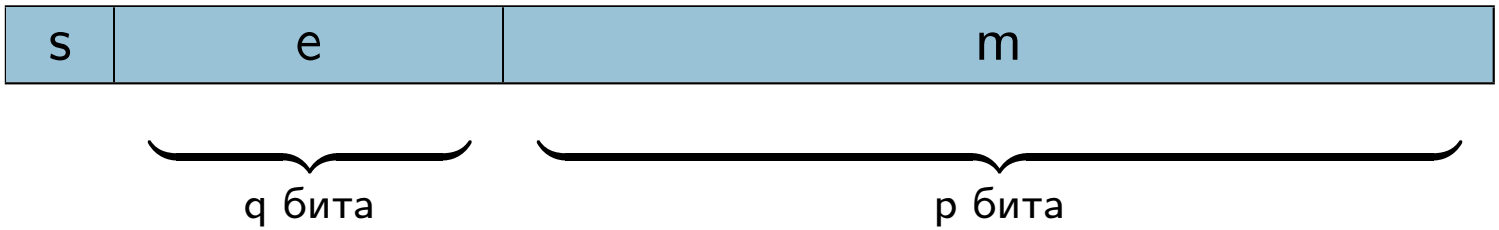
- Внимание: това не са реални числа!
  - А какво са реални числа?
- Още ще ги наричаме и **дробни числа**
- Представяне в паметта
  - $f = (-1)^s \cdot m \cdot 2^e$



  - $s \in \{0, 1\}$ — знак
  - $m \in [0; 2^{23} - 1]$ — мантиса
  - $e \in [-126; 127]$ — експонента
  - машинна нула: $(-2^{-127}; 2^{-127})$

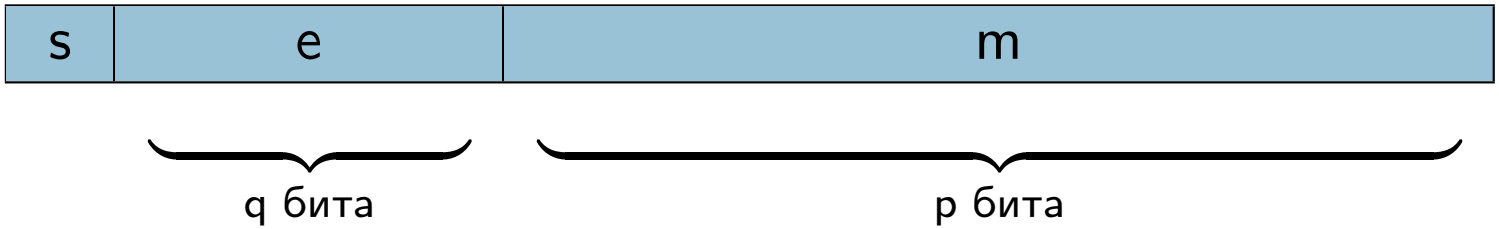
Общо представяне на числа с плаваща запетая

$$f = (-1)^s \cdot m \cdot 2^e$$



- $s \in \{0, 1\}$ — знак
- $m \in [0; 2^p - 1]$ — мантиса
  - p — точност, брой двоични цифри на мантисата
- $e \in [e_{min}; e_{max}]$ — експонента
  - $e_{min} = -e_{max} + 1$
  - $e_{max} = 2^{q-1} - 1$
  - q — обхват на експонентата
  - $e_{min} - 1$ и $e_{max} + 1$ са запазени за служебно ползване
- общо използвани битове: $p + q + 1$
- машинна нула: $(-2^{e_{min}-1}; 2^{e_{min}-1})$

Типове дробни числа



тип	размер	точност (p)	обхват (q)
`float`	4 байта = 32 бита	23 бита	8 бита
`double`	8 байта = 64 бита	52 бита	11 бита
`long double`	16 байта = 128 бита	112 бита	15 бита

Дробни литерали

- [<цяло_число>].[<цяло_без_знак>][(E|e)<цяло_число>]
- Примери: `1`, `2.34`, `12e-2`, `10.14E+03`, `.23`
- Операции:
  - всички за целочислен тип без %
  - / е дробно деление, а не частно!
  - сравненията == и != са ненадеждни!

Математически функции

```
#include <cmath>
```

- `abs(x)`, `fabs(x)`
- `sin(x)`, `cos(x)`, `tan(x)`, `asin(x)`, `acos(x)`, `atan(x)`
- `exp(x)`, `log(x)`, `log10(x)`
- `ceil(x)`, `floor(x)`
- `sqrt(x)`, `pow(x, n)`

Преобразуване на типове

- `bool` → `char` → `short` → `int` → `long` → `float` → `double`
- `unsigned char` → `unsigned short` → `unsigned` → `unsigned long`
- обратната посока може да доведе до загуба на информация
- експлицитно преобразуване на типове:
  <преобразуване> ::= (<тип>)<израз>

Приоритет на операциите

1. Обръщения към функции
2. Скоби
3. !, +, - (едноместни)
4. *, /, %
5. +, - (двуместни)
6. <<, >>
7. <, <=, >, >=
8. ==, !=
9. &&
10. ||