

Imperatív programozás

Kifejezések

Kozsik Tamás és mások

Eötvös Loránd Tudományegyetem

2022. augusztus 20.



Tartalomjegyzék

- 1 Lexika
- 2 Szintaktika
- 3 Szemantika
- 4 Kifejezések kiértékelése

Példák

`n + 1`

`3.14 * r * r`

`3 * v[0]`

`x < 3.14`

`3 * (r1 + r2) == factorial(x)`



Lexikális elemek

- Literálok
- Operátorok
- Azonosítók
- Zárójelek
- Egyéb jelek, pl.: vessző

Függvényhívás szintaxisa

```
<function-call> ::= <identifier>()  
                  | <identifier>(<argument-list>)
```

```
<argument-list> ::= <expression>  
                   | <expression>, <argument-list>
```

pi()

factorial(n + m)

min(0, x + y)



Operátorok arítása

- Unáris, pl.: $-x$, $c++$
- Bináris, pl.: $x - y$
- Ternáris, pl.: $x < 0 ? 0 : x$

Operátorok fixitása

- Prefix, pl.: ++c
- Postfix, pl.: c++
- Infix, pl.: $x + y$
- Mixfix, pl.: $x < 0 ? 0 : x$



Értelmes kifejezések

- A benne szereplő azonosítók deklaráltak (Önmagában az $n + 1$ nem értelmes)
- Jól típusozott (Van értelmük C-ben: $3 + 3.14$, "hello" + 1, nincs értelme C-ben: "hello" * 42)
- Típushelyes, de attól még nincs értelme (pl.: $1 / 0$)



1. *Journal of Management Studies*, 1997, 34, 1, 1-14.

1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53. 54. 55. 56. 57. 58. 59. 60. 61. 62. 63. 64. 65. 66. 67. 68. 69. 70. 71. 72. 73. 74. 75. 76. 77. 78. 79. 80. 81. 82. 83. 84. 85. 86. 87. 88. 89. 90. 91. 92. 93. 94. 95. 96. 97. 98. 99. 100. 101. 102. 103. 104. 105. 106. 107. 108. 109. 110. 111. 112. 113. 114. 115. 116. 117. 118. 119. 120. 121. 122. 123. 124. 125. 126. 127. 128. 129. 130. 131. 132. 133. 134. 135. 136. 137. 138. 139. 140. 141. 142. 143. 144. 145. 146. 147. 148. 149. 150. 151. 152. 153. 154. 155. 156. 157. 158. 159. 160. 161. 162. 163. 164. 165. 166. 167. 168. 169. 170. 171. 172. 173. 174. 175. 176. 177. 178. 179. 180. 181. 182. 183. 184. 185. 186. 187. 188. 189. 190. 191. 192. 193. 194. 195. 196. 197. 198. 199. 200. 201. 202. 203. 204. 205. 206. 207. 208. 209. 210. 211. 212. 213. 214. 215. 216. 217. 218. 219. 220. 221. 222. 223. 224. 225. 226. 227. 228. 229. 230. 231. 232. 233. 234. 235. 236. 237. 238. 239. 240. 241. 242. 243. 244. 245. 246. 247. 248. 249. 250. 251. 252. 253. 254. 255. 256. 257. 258. 259. 260. 261. 262. 263. 264. 265. 266. 267. 268. 269. 270. 271. 272. 273. 274. 275. 276. 277. 278. 279. 280. 281. 282. 283. 284. 285. 286. 287. 288. 289. 290. 291. 292. 293. 294. 295. 296. 297. 298. 299. 300. 301. 302. 303. 304. 305. 306. 307. 308. 309. 310. 311. 312. 313. 314. 315. 316. 317. 318. 319. 320. 321. 322. 323. 324. 325. 326. 327. 328. 329. 330. 331. 332. 333. 334. 335. 336. 337. 338. 339. 340. 341. 342. 343. 344. 345. 346. 347. 348. 349. 350. 351. 352. 353. 354. 355. 356. 357. 358. 359. 360. 361. 362. 363. 364. 365. 366. 367. 368. 369. 370. 371. 372. 373. 374. 375. 376. 377. 378. 379. 380. 381. 382. 383. 384. 385. 386. 387. 388. 389. 390. 391. 392. 393. 394. 395. 396. 397. 398. 399. 400. 401. 402. 403. 404. 405. 406. 407. 408. 409. 410. 411. 412. 413. 414. 415. 416. 417. 418. 419. 420. 421. 422. 423. 424. 425. 426. 427. 428. 429. 430. 431. 432. 433. 434. 435. 436. 437. 438. 439. 440. 441. 442. 443. 444. 445. 446. 447. 448. 449. 450. 451. 452. 453. 454. 455. 456. 457. 458. 459. 460. 461. 462. 463. 464. 465. 466. 467. 468. 469. 470. 471. 472. 473. 474. 475. 476. 477. 478. 479. 480. 481. 482. 483. 484. 485. 486. 487. 488. 489. 490. 491. 492. 493. 494. 495. 496. 497. 498. 499. 500. 501. 502. 503. 504. 505. 506. 507. 508. 509. 510. 511. 512. 513. 514. 515. 516. 517. 518. 519. 520. 521. 522. 523. 524. 525. 526. 527. 528. 529. 530. 531. 532. 533. 534. 535. 536. 537. 538. 539. 540. 541. 542. 543. 544. 545. 546. 547. 548. 549. 550. 551. 552. 553. 554. 555. 556. 557. 558. 559. 560. 561. 562. 563. 564. 565. 566. 567. 568. 569. 570. 571. 572. 573. 574. 575. 576. 577. 578. 579. 580. 581. 582. 583. 584. 585. 586. 587. 588. 589. 590. 591. 592. 593. 594. 595. 596. 597. 598. 599. 600. 601. 602. 603. 604. 605. 606. 607. 608. 609. 610. 611. 612. 613. 614. 615. 616. 617. 618. 619. 620. 621. 622. 623. 624. 625. 626. 627. 628. 629. 630. 631. 632. 633. 634. 635. 636. 637. 638. 639. 640. 641. 642. 643. 644. 645. 646. 647. 648. 649. 650. 651. 652. 653. 654. 655. 656. 657. 658. 659. 660. 661. 662. 663. 664. 665. 666. 667. 668. 669. 670. 671. 672. 673. 674. 675. 676. 677. 678. 679. 680. 681. 682. 683. 684. 685. 686. 687. 688. 689. 690. 691. 692. 693. 694. 695. 696. 697. 698. 699. 700. 701. 702. 703. 704. 705. 706. 707. 708. 709. 710. 711. 712. 713. 714. 715. 716. 717. 718. 719. 720. 721. 722. 723. 724. 725. 726. 727. 728. 729. 730. 731. 732. 733. 734. 735. 736. 737. 738. 739. 740. 741. 742. 743. 744. 745. 746. 747. 748. 749. 750. 751. 752. 753. 754. 755. 756. 757. 758. 759. 760. 761. 762. 763. 764. 765. 766. 767. 768. 769. 770. 771. 772. 773. 774. 775. 776. 777. 778. 779. 780. 781. 782. 783. 784. 785. 786. 787. 788. 789. 790. 791. 792. 793. 794. 795. 796. 797. 798. 799. 800. 801. 802. 803. 804. 805. 806. 807. 808. 809. 810. 811. 812. 813. 814. 815. 816. 817. 818. 819. 820. 821. 822. 823. 824. 825. 826. 827. 828. 829. 830. 831. 832. 833. 834. 835. 836. 837. 838. 839. 840. 84



Kiértékelés szabályai

- Teljesen bezárójelezett kifejezés
 $3 + ((12 - 3) * 4)$
- Precedencia: $a * b$ erősebben köt, mint $a + b$
 $12 - 3 * 4$
- Bal- és jobbasszociativitás
 - Azonos precedenciaszintű operátorok esetén
 - $3 * n / 2$ jelentése $(3 * n) / 2$ (bal-asszociatív op.)
 - $n = m = 1$ jelentése $n = (m = 1)$ (jobb-asszociatív op.)



Értékadás

Értékadó utasítás

```
n = 1;
```

Mellékhatásos kifejezés

```
n = 1
```

Mellékhatásos kifejezés értéke

```
(n = 1) értéke 1
```

Érték továbbgyűrzése

```
m = (n = 1)
```



```
printf("%d", n)
n = 1
i *= j
i++
++i
```

Eggyel növelő/csökkentő operátorok

```
int n = 5;
```

```
int i;
```

```
i = ++n;
```

```
n == 6
```

```
i == 6
```

```
int m = 5;
```

```
int j;
```

```
j = m++;
```

```
m == 6
```

```
j == 5
```



Lustaság, mohóság

- Mohó: az $A + B$ alakú kifejezés
- Lusta: az $A \ \&\& \ B$ alakú kifejezés (továbbá: $||$, illetve $?:$)

```
int f() { printf("f"); return 1; }  
int g() { printf("g"); return 2; }
```

```
int i = f() + g();  
/* Kimenet: fg, Eredmény: i == 3 */
```

```
bool b1 = (f() == 0) && (g() == 2);  
/* Kimenet: f, Eredmény: b1 == false */
```

```
bool b2 = (f() == 1) || (g() == 3);  
/* Kimenet: f, Eredmény: b2 == true */
```

Lusta/mohó logikai éselés értéke

Jelölje \uparrow , \downarrow , \perp és ∞ a négy lehetséges eredményt egy logikai kifejezés kiértékeléséhez: igaz, hamis, kivétel, nem termináló számítás. Az $\alpha \wedge \beta$ kifejezés értéke az α és β értékének függvényében:

| $\alpha \wedge_{\text{lusta}} \beta$ | $\beta = \uparrow$ | $\beta = \downarrow$ | $\beta = \perp$ | $\beta = \infty$ |
|--------------------------------------|--------------------|----------------------|-----------------|------------------|
| $\alpha = \uparrow$ | \uparrow | \downarrow | \perp | ∞ |
| $\alpha = \downarrow$ | \downarrow | \downarrow | \downarrow | \downarrow |
| $\alpha = \perp$ | \perp | \perp | \perp | \perp |
| $\alpha = \infty$ | ∞ | ∞ | ∞ | ∞ |

| $\alpha \wedge_{\text{mohó}} \beta$ | $\beta = \uparrow$ | $\beta = \downarrow$ | $\beta = \perp$ | $\beta = \infty$ |
|-------------------------------------|--------------------|----------------------|-----------------|------------------|
| $\alpha = \uparrow$ | \uparrow | \downarrow | \perp | ∞ |
| $\alpha = \downarrow$ | \downarrow | \downarrow | \perp | ∞ |
| $\alpha = \perp$ | \perp | \perp | \perp | \perp |
| $\alpha = \infty$ | ∞ | ∞ | ∞ | ∞ |



Példa

Keressük meg egy tömb első negatív elemének indexét

```
/* Jó */  
for (i = 0; i < LENGTH && array[i] >= 0; ++i);  
/* Rossz */  
for (i = 0; array[i] >= 0 && i < LENGTH; ++i);
```

Mellékhatalás a lusta operátor operandusaiban

```
(n = 1) + (m = 1)  
(n = 1) || (m = 1)
```



Operandusok, függvényparaméterek kiértékelési sorrendje

Kiértékelési sorrend nem specifikált

```
int f() { printf("f"); return 1; }
```

```
int g() { printf("g"); return 2; }
```

```
void printSum(int a, int b) { printf("%d\n", a + b); }
```

```
printf("%d\n", f() + g());           /* Kimenet: fg3 vagy gf3 */
```

```
printSum(f(), g());                 /* Kimenet: fg3 vagy gf3 */
```

Ilyet ne csináljunk!

```
int i = 2;
```

```
int j = i-- - --i;
```

```
printf("%d\n", j);                   /* Kimenet: 2 vagy 0 */
```



Szekvenciapont

- Teljes kifejezés végén
- Függvényhívás aktuális paraméterlistájának kiértékelése végén
- Lusta operátorok első operandusának kiértékelése után
- Vessző operátornál



Vessző-operátor

```
<expression> ::= ...  
                | <expression>, <expression>
```

- Az eredménye a jobboldali kifejezés eredménye
- Alacsony precedenciaszintű

```
int f() { printf("f"); return 1; }  
int g() { printf("g"); return 2; }  
int h() { printf("h"); return 3; }
```

```
printf("%d\n", (f(), g(), h()));      /* Kimenet: fgh3 */
```

