Додавання довгих чисел з різними знаками зводиться до додавання чисел, якщо обидва числа з однаковими знаками, або до віднімання довгих чисел. В другому випадку потрібно спочатку порівняти числа і завжди віднімати від більшого числа менше, додавши до нього відповідний знак.

**Додавання довгих чисел**

При додаванні «у стовпчик» двох чисел з різною кількість, числа записуються таким чином, щоб останні цифри знаходилися одна під одною. Якщо ми зберігаємо число у масиві, то нам досить незручно пересувати число з меншою кількістю знаків таким чином, щоб молодші розряди стояли на однакових позиціях. Крім того результат додавання може складатися з більшої кількості цифр, ніж кожен з доданків. У зв’язку з цим нам може знадобитися записати цифру в початок (тобто першою). Так як перша цифра записана у комірці з номером 1, то доведеться зсунути все число на одну позицію вправо і тільки після цього записати старшу цифру. У такому випадку реалізація стає дуже заплутаною з великою кількістю випадків и частими зсуненнями числа по масиву. Все це може призвести до уповільнення програми та великої кількості помилок при реалізації. Щоб уникнути таких труднощів, краще за все зберігати число у «перевернутому» вигляді, тобто, у першій комірці масиву буде записана молодша цифра (одиниці), у другій будуть записані десятки і так далі. Це вирішує відразі всі перераховані проблеми. По перше, всі числа тепер записані таким чином, що їхні останні цифри записані одна під одною, по друге, для додавання до числа цифри зліва, необхідно просто дописати її у кінець масиву.

Реалізація додавання на мові Pascal.

**const**

 DMAX = 100;

**type**

 thuge = array [0..DMAX] of integer;

**procedure** add(**var** a, b : thuge);

{функція додає до числа a число b}

**var**

 i, r : integer;

{r – означає залишок}

**begin**

 **if** a[0] < b[0] **then** a[0] := b[0];

{додавати необхідно до розміру більшого числа}

 r := 0;

{при додаванні молодших цифр залишок у нас 0}

 **for** i := 1 **to** a[0] **do**

 **begin**

 a[i] := a[i] + b[i] + r;

 {сума чергових цифр і перенесення}

 **if** a[i] >= 10 **then**

 {випадок, коли виникає перенесення у наступний розряд}

 **begin**

 r := 1;

 dec(a[i], 10);

 **end** **else** **begin**

 {випадок, коли перенесення не відбувається}

 r := 0;

 **end**;

 **end**;

{якщо після додавання залишилось ще перенесення, то необхідно додати ще одну цифру}

 **if** r > 0 **then**

 **begin**

 inc(a[0]);

 a[a[0]] := r;

 **end**;

**end**;

Реалізація додавання на мові С++.

#define DMAX 100

**typedef** int thuge[DMAX];

**void** add(thuge &a, thuge &b){

// функція додає до числа a число b

 **if** (a[0] < b[0]) a[0] = b[0];

// додавати необхідно до розміру більшого числа

 int r = 0;

/\* r – означає залишок

при додаванні молодших цифр залишок у нас 0\*/

 **for**(int i = 1; i <= a[0]; ++i) {

 a[i] += b[i] + r;

 // сума чергових цифр і перенесення

 **if** (a[i] >= 10) {

 // випадок, коли виникає перенесення у наступний розряд

 r = 1;

 a[i] -= 10;

 } **else** {

 // випадок, коли перенесення не відбувається

 r = 0;

 }

 }

 // якщо після додавання залишилось ще перенесення, то необхідно додати ще одну цифру

 **if** (r > 0) {

 a[0]++;

 a[a[0]] = r;

 }

}

 Існують різні системи числення. Взагалі, ми можемо працювати у будь-якій з них. Але необхідно пам’ятати, що перенесення відбувається, якщо сума стала більшою чи дорівнювати основі системи числення. Для роботи у системі числення *base*, необхідно замінити у програмі всі *10* на *base*. (У такому випадку бажано завести константу *base*.) Для досягнення максимальної швидкості роботи програми необхідно обирати основу системи числення, яка буде дорівнювати деякому степеню двійки. При такому виборі можна реалізувати більш швидкі операції, пов’язані з *base*. Але в такому випадку виникають труднощі при введенні та виведенні числа. Для введення та виведення доведеться переводити число з однієї системи числення в іншу. Це достатньо трудомістка операція, тому для нас будуть цікавими системи числення з основою 10K (для деякого натурального числа K). Така система числення, фактично, означає, що в одній комірці масиву буде зберігатися не одна, а K цифр. Очевидно, що така операція додавання буде працювати в K раз швидше, так як в числах буде в K разів менше цифр.

**Віднімання довгих чисел**

Віднімання у стовпчик практично аналогічне. Якщо при відніманні результат виявляється від’ємним, то ми позичаємо одиницю із старшого розряду. Ми будемо працювати лише з невід’ємними числами, тому при відніманні будемо мати на увазі, що різниця невід’ємна. На відміну від додавання кількість цифр у різниці може зменшитися, тому буде необхідно перерахувати кількість цифр.

**Реалізація віднімання на мові Pascal.**

**procedure** subtract(**var** a, b : thuge);

{функція віднімає від числа a число b}

**var**

 i, r : integer;

{r - означає, чи була позика одиниці з поточного розряду}

begin

 r := 0;

{позика з молодшого розряду відсутня}

 **for** i := 1 **to** a[0] **do**

 **begin**

 a[i] := a[i] - b[i] - r;

 {різниця чергових цифр з урахуванням позики}

 **if** a[i] < 0 **then**

 {випадок, коли відбувається позика з наступного розряду}

 **begin**

 r := 1;

 inc(a[i], base);

 **end else begin**

 {випадок, коли позики не відбувається}

 r := 0;

 **end**;

 **end**;

 {Різниця може мати меншу кількість цифр,

 Тому необхідно у разі потреби зменшити кількість цифр}

 **while** (a[0] > 1) **and** (a[a[0]] = 0) **do**

 **begin**

 dec(a[0]);

 **end**;

**end**;

**Реалізація віднімання на мові С++.**

**void** subtract(thuge &a, thuge &b) {

// функція віднімає від числа a число b

// r - означає, чи була позика одиниці з поточного розряду

 int r = 0;

// позика з молодшого розряду відсутня

 **for**(int i = 1; i <= a[0]; ++i) {

 a[i] -= b[i] + r;

 // різниця чергових цифр з урахуванням позики **if** (a[i] < 0) {

 // випадок, коли відбувається позика з наступного розряду

 r = 1;

 a[i] += base;

 } **else** {

 // випадок, коли позики не відбувається

 r = 0;

 }

 }

 /\* Різниця може мати меншу кількість цифр,

 Тому необхідно у разі потреби зменшити кількість цифр\*/

 **while** (a[0] > 1 && a[a[0]] == 0) {

 --a[0];

 }

}

 Додавання та віднімання працює за довжину найбільшого числа. Числа, котрі зустрічаються у процесі обчислень, не перевищують по модулю *2 \* base*. Отже в одній комірці масиву при використанні типу longint (int) можна зберігати до 9 десяткових цифр (тобто основа системи числення може бути до 109).

**Порівняння чисел**

 Перше, на що ми звернемо увагу, коли будемо порівнювати числа, буде кількість цифр у них. Якщо кількість цифр різна, то більше те з них, котре має більше цифр. Якщо кількість цифр однакова, то необхідно порівнювати , починаючи зі старшої цифри. При знаходженні першої невідповідності(тобто найстарша цифра, у якій є відмінність), можна визначити яке з чисел є більшим. Якщо два числа не мають відмінностей, то вони.

*Pascal*

**function** compare(**var** a, b : thuge) : integer;

{

a < b => -1

a > b => 1

a = b => 0

}

**var**

 i : integer;

**begin**

{порівняння за кількістю цифр}

 **if** a[0] < b[0] **then**

 **begin**

 compare := -1;

 **exit**;

 **end**;

 **if** a[0] > b[0] **then**

 **begin**

 compare := 1;

 **exit**;

 **end**;

{порівняння у випадку однакової кількості цифр}

 **for** i:= a[0] **downto** 1 **do**

 **begin**

 **if** a[i] < b[i] **then**

 **begin**

 compare := -1;

 **exit**;

 **end**;

 **if** a[i] > b[i] **then**

 **begin**

 compare := 1;

 **exit**;

 **end**;

 **end**;

 compare := 0;

**end**;

*С++*

int **compare**(thuge &a, thuge &b) {

/\*

a < b => -1

a > b => 1

a = b => 0

\*/

//порівняння за кількістю цифр

 **if** (a[0] < b[0]) {

 **return** -1;

 }

 **if** (a[0] > b[0]) {

 **return** 1;

 }

/\* порівняння у випадку однакової кількості цифр \*/

 **for**(int i = a[0]; i >= 1; --i) {

 **if** (a[i] < b[i]) {

 **return** -1;

 }

 **if** (a[i] > b[i]) {

 **return** 1;

 }

 }

 **return** 0;

}

**Введення та виведення довгого числа**

 Так як введений нами тип thuge не є стандартним, то його неможливо прочитати, використовуючи read та виводити за допомогою функції write. Для того, щоб вивести число, що записане у десятковій системі достатньо послідовно вивести цифри, починаючи зі старшої. Якщо число записане у системі числення 10K, то це не зовсім вірно. Число 109 в 100-ричной системі числення складається з двох цифр: 1 и 9. Тому необхідно бути уважним при виведенні цифр, що складаються з невеликої кількості цифр, так як до таких цифр при виведенні необхідно дописати лідируючі нулі. Зверніть увагу, що до самої першої цифри дописувати нулі не потрібно.

*Pascal*

**const** digit = 4;

{кількість цифр в одній комірці масиву}

**procedure** writeHuge(**var** a : thuge);

**var**

 i : integer;

 s : string;

**begin**

 write(a[a[0]]);

 **for** i:= a[0] - 1 **downto** 1 **do**

 **begin**

 str(a[i], s);

 {дописуємо нулі зліва при необхідності}

 **while** length(s) < digit **do** s := '0' + s;

 write(s);

 **end**;

 writeln;

**end**;

*C++*

**const** char \* digitFormat = "%.4d";

/\*формат виводу числа рівно з 4 цифр,

цифри, яких бракує, доповнюються нулями\*/

**void** writeHuge(thuge &a) {

 printf("%d", a[a[0]]);

 **for**(int i = a[0] - 1; i >= 1; --i) {

 printf(digitFormat, a[i]);

 }

 printf("\n");

}

При читанні довгого числа необхідно пам’ятати, що цифри записуються у зворотному порядку. Але у випадку системи числення більшої за 10, «десяткові цифри» всередині цифри більшої системи числення йдуть у прямому порядку. Наприклад, число 12345678 в 100-ричній системі числення буде записане як: 78, 56, 34, 122.

*Pascal*

**procedure** readHuge(**var** a : thuge);

**var**

 i, j, pos : integer;

 s : string;

**begin**

 read(s); //зчитування рядка

 fillchar(a, **sizeof**(a), 0);

 a[0] := (length(s) - 1) div digit + 1;

 {знаходження кількості цифр в потрібній системі числення}

 **for** i:= 1 **to** a[0] **do**

 **begin**

 {знаходження і–тої цифри}

 **for** j:= digit - 1 **downto** 0 **do**

 **begin**

 pos := length(s) - (i - 1) \* digit - j;

 {знаходження позиції, з якої необхідно дописати до i-тої цифри}

 **if** pos > 0 **then**

­­­­­­­­­­­ a[i] := a[i] \* 10 + ord(s[pos]) - ord('0');

 **end**;

 **end**;

**end**;

*C++*

**const** int digit = 4;

/\*кількість цифр в одній комірці масиву\*/

**void** readHuge(thuge &a) {

 char s[DMAX \* digit + 1];

 scanf("%s", s);//чтение строки

 memset(a, 0, **sizeof**(a));

 int len = strlen(s);

 a[0] = (len - 1) / digit + 1;

 /\*знаходження кількості цифр в потрібній системі числення\*/

 **for**(int i = 1; i <= a[0]; ++i) {

 /\*знаходження і-тої цифри\*/

 **for**(int j = digit; j >= 1; --j) {

 int pos = len - (i - 1) \* digit - j;

 /\* знаходження позиції, з якої необхідно дописати до i-тої цифри. Пам’ятайте, що масив s індексовано з 0\*/

 **if** (pos >= 0) {

 a[i] = a[i] \* 10 + (s[pos] - '0');

 }

 }

 }

}