Аритметика с дълги числа

Стоян Капралов

В лекцията има нови алгоритмични и технологични моменти.

В много задачи от Националната олимпиада по информатика и националните състезания се налага да се използва аритметика с дълги числа.

Има различни алгоритмични и програмни реализации.

Един възможен вариант е представен в първата част на книгата „Състезателно програмиране за група D“ с автор Красимир Манев. За представяне на абстрактния тип *голямо цяло* се използват модифицирани C++ низове.

За нашата реализация сме избрали представяне на дългите числа във вектори от `int`.

За удобство дефинираме псевдоним на `vector<int>`:

```
using num = vector<int>;
```

С този ред казваме на компилатора: „*Отсега нататък, вместо да пиша* `vector<int>`*, искам да използвам просто думата* `num`“.

Следните два записа са напълно аналогични:

```
// Стандартният начин:
std::vector<int> a;
```

```
// Начинът с псевдонима num:
num a;
```

В променливи от тип `num` ще съхраняваме цели неотрицателни числа с произволен брой цифри в десетична бройна система.

Пример. Числото $27501_{(10)} = 2*10^4 + 7*10^3 + 5*10^2 + 0*10^1 + 1.10^0$ записваме без водещи нули цифра по цифра в променливата `num a;` като `a[k]` е коефициентът пред 10^k.

	4	3	2	1	0
a	2	7	5	0	1

Декларираме често използваните константи `0` и `1`:

```
const num ZERO{0}, ONE{1};
```

„Общуването“ с външния свят става посредством обекти от класа `string`, като за целта използваме следните две функции:

```
num to_num(string s)
{ int n = s.size();
  num a(n);
  for(int i=0,j=n-1; i<n; i++,j--)
    a[i] = s[j] - '0';
  return a;
}

string to_string(num a)
{ int n = a.size();
  string s(n,'0');
  for(int i=0,j=n-1; i<n; i++,j--)
    s[i] += a[j];
  return s;
}
```

Това веднага ни позволява да предефинираме операциите за въвеждане от поток и за извеждане в поток.

```
istream& operator>>(istream& cin, num& a)
{ string s; cin >> s;
  a = to_num(s);
  return cin;
}

ostream& operator<<(ostream& cout, const num& a)
{ cout << to_string(a);
  return cout;
}
```

Вече може да пишем

```
cin >> a;
```

и

```
cout << a;
```

и ефектът е същият като

```
string s; cin >> s; num a=to_num(s);
```

и съответно

```
cout << to_string(a);
```

Следват функците за събиране и изваждане на дълги числа, които реализират обичайните алгоритми за събиране и изваждане на многоцифрени числа.

```
num add(num a, num b)
{ while(a.size() < b.size()) a.push_back(0);
  while(b.size() < a.size()) b.push_back(0);

  int n = a.size();
  num x(n);
  int c = 0;
  for(int i=0; i<n; i++)
  { x[i] = a[i] + b[i] + c;
    c = x[i] / 10;
    x[i] = x[i] % 10;
  }
  while(c>0)
  { x.push_back(c%10);
    c = c/10;
  }
  return x;
}
```

Предефинираме операциията + за работа с дълги числа

```
num operator+(num a, num b)
{ return add(a,b); }
```

```
num sub(num a, num b)
// suppose a>=b
{ int n = a.size();
  while(b.size()<n) b.push_back(0);
  while(b.size()>n) b.pop_back();

  num x(n);
  int c = 0;
  for(int i=0; i<n; i++)
  { x[i] = a[i] - b[i] - c;
    if(x[i]>=0) c=0;
    else
    { c=1; x[i] += 10; }
  }
  while(x.size()>1 and x.back()==0) x.pop_back();
  return x;
}
```

Предефинираме и операцията минус.

```
num operator-(num a, num b)
{ return sub(a,b); }
```

Следва функцията за умножение на дълги числа.

Произведението `a[i]*b[j]` съответства на умножението $10^{i}*10^{j}=10^{i+j}$ и следователно трябва да се отрази в елемента `x[i+j]` от резултата.

```
num mul(num a, num b)
{ int m = a.size();
  int n = b.size();
  num x(m+n);
  for(int i=0; i<m; i++)
    for(int j=0; j<n; j++)
      x[i+j] = x[i+j] + a[i]*b[j];
  int c = 0;
  for(int i=0; i<m+n; i++)
  { x[i] = x[i] + c;
```

```
    c = x[i] / 10;
    x[i] = x[i] % 10;
  }
  while(c>0)
  { x.push_back(c%10);
    c = c / 10;
  }
  while(x.size()>1 and x.back()==0)
    x.pop_back();
  return x;
}

num operator*(num a, num b)
{ return mul(a,b); }
```

Алгоритмите за деление и намиране на остатък вървят „ръка за ръка“.

Най-напред да разгледаме функцията

`pair<num,int> divmod(num a, int x),`

която по дадени дълго цяло число `a` и едноцифрено число `x` връща като двойка целочисленото частно и остатъка при делението на `a` на `x`. Реализиран е училищният алгоритъм за деление на едноцифрено число. За разбирането на алгоритъма си мислете, че `x` наистина е едноцифрено число, макар че резултатите са коректни за всякакви не много големи числа от тип `int`.

```
pair<num,int> divmod(num a, int x)
{ if(x==1) return pair<num,int>{a,0};

  int n=a.size();
  num b(n);
  int p=0;
  for(int i=n-1; i>=0; i--)
  { p = p*10 + a[i];
    b[i] = p/x;
    p = p % x;
  }
```

```
  while(b.size()>1 and b.back()==0) b.pop_back();
  return pair<num,int>{b,p};
}

num operator/(num a, int x)
{ return divmod(a,x).first; }

int operator%(num a, int x)
{ return divmod(a,x).second; }
```

За намиране на частното `x` при делението на дългото число `b` на дългото число `a`, разглеждаме неравенството `a*x ≤ b`. Търсеното число `x` е най-голямото цяло число, което е решение на неравенството.

Това позволява да намерим `x` с двоично търсене. Поддържаме интервал `[x, y)`, такъв че `x` е решение на неравенството, а `y` не е, т.е. `a*x ≤ b`, но `a*y > b`.
На всяка стъпка намираме средата `m` на интервала `num m = x+(y-x)/2;`
и продължаваме с лявата или дясната половина, в зависимост от ситуацията.

Когато разликата `y-x` стане 1, числото `x` е търсеното.

Сложността на този алгоритъм за деление е O(log b).

Важна забележка: при намирането на `m` използваме деление на едноцифрено число.

```
num div(num b, num a)
{ num x = ZERO;
  num y = b+ONE;

  while(y-x != ONE)
  { num m = x+(y-x)/2;
    num s = a*m;
    if(s<=b) x = m;
    else y = m;
  }
  return x;
}
```

```
num operator/(num a, num b)
{ return div(a,b); }
```

Щом вече разполагаме с деление на дълги числа, намирането на остатъка не е проблем.

```
num mod(num a, num b)
{ return a - a/b*b; }
```

Съответно, предефинираме и операцията `%` за работа с дълги числа.

```
num operator%(num a, num b)
{ return mod(a,b); }
```

Накрая да разгледаме предефиниране на операцията за сравнение „по-малко“.

```
bool operator<(num a, num b)
{ while(a.size() < b.size()) a.push_back(0);
  while(b.size() < a.size()) b.push_back(0);
  return to_string(a) < to_string(b);
}
```

Сега можем да сортираме вектор от дълги цели числа `vector<num> a;` по обичайния начин `sort(a.begin(),a.end());` без да задаваме функция за сравнение.

Да разгледаме едно приложение на нашата библиотека за аритметика с дълги числа, като напишем решение на задача от Националната олимпиада по информатика (а също и от контролното състезание за 6 клас на Националната школа по информатика, Габрово, 2026).

```
/// НОИ-2, 2018, група D, sdnf
int main()
{ int n; cin >> n;
  num f = ONE; // f = n!
  for(int k=2; k<=n; k++)
    f = f * k;

  int sum=0;
  for(int x:f) sum = sum + x;
  cout << sum << endl;

  return 0;
}
```