Чевардин А.Д.
Научный руководитель: к.т.н., зав. каф. ПМ и САПР Котов В.В.
Ковровская государственная технологическая академия имени В.А. Дегтярева
Ковров, Маяковского, 19
aleksdch@yandex.ru

Реализация алгоритмов длинной арифметики

В нашей работе были реализованы несколько вариантов реализации алгоритмов длинных чисел на языке программирования С++. Цель данного проекта заключается в том, чтобы наглядно показать различие методов, а также чтобы каждый выбрал понравившейся ему способ. Алгоритмы создавались для следующих действий: сложение и вычитание.

Работа была выполнена на языке программированияС++ в среде программирования Code::Blocks. Все алгоритмы были протестированы и отлажены. Всего было представлено 4 программы – 3 на сложение и 1 на вычитание.

Длина арифметика – алгоритмы, позволяющие производить математические действия с длинными числами. Длинные числа – числа, выходящие за диапазон, который поддерживает переменная. У каждой переменной свой диапазон.

Длинная арифметика – способ работы с числами, не входящими в диапазон переменной, то есть обход диапазонного ограничения.

Сложение было реализовано следующими способами:

Одномерные массивы с ручным выбором длины числа (ориентация по длиннейшему)

```
#include <iostream>
using namespace std;
int main()
{
   int m;
   cin>>m; ← Задача
   int a[m]; размера
   int b[m]; большего
   int c[m]; числа

   for(int i=0; i<m; i++)
   {                Ввод первого
      cin>>a[i]; числа
   }                (большего)
   cout<<"\n+\n";
   for(int i=0; i<m; i++)
   {                Ввод второго
      cin>>b[i]; числа
   }                (меньшего)
int k=0;
   for(int i=0; i<m; i++)
   {
      if(b[i]==11) //Подсчёт
      {            количества
         k=k+1;    чисел 11
      }            (пустых мест)
   }
   int counter=0;
   while(counter<k-1)
   {
      for(int i=0; i<m; i++)
      {
         if(b[i]==11 && b[i-1]!=11)
         {
            int c;      //Перемещен
            c=b[i];     ие всех чисел
            b[i]=b[i-1]; 11 с конца в
            b[i-1]=c;   начало
      }
   }
counter=counter+1;
   }
   for(int i=0; i<m; i++)
   {
      if(b[i]==11)
      {
         b[i]=0;  //Замена
      }           чисел 11 на 0
   }
    int k2=0;
      int j=0;
      for(int i=m-1; i>=0; i--)
      {
         c[i]=a[i]+b[i];  //Суммирование
         if(c[i]>9)
         {
            int g;
            g=c[i];
            c[i]=g/10;
            j=g%10;
            a[i-1]=a[i-1]+j;
         }
         if(c[i]<=9)
         {
            c[i]=c[i];
            a[i-1]=a[i-1];
         }
         k2=k2+1;
      }
```

Рис. 1. Фрагмент кода

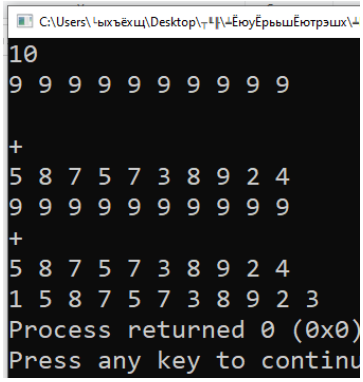

Рис. 2. Результат работы программы.

Одномерные массивы с автоматическим выбором длины числа

```
if (size_a > size_b)
   length = size_a + 1;  // определение длины массива результата

else
   length = size_b + 1;

for (int i = 0; i < length; i++)
{
   b[i] += a[i]; // суммирование последних разрядов чисел
   b[i + 1] += (b[i] / 10); // если есть разряд для переноса, переносим его в следующий разряд
   b[i] %= 10; // и отсекаем его
}

if (b[length - 1] == 0)
   length--;
```

Рис. 3. Часть программы

С помощью строк и векторного (динамического) массива.

```
#include <iostream>
#include <sstream>
#include <vector>
#define max9 999999999
using namespace std;
int main()
{
   stringstream ss;
   string num;
   int m;
   string sa;
   size_t pos = 0;
   vector<unsigned int> a(1);
   a.clear();
   getline(cin,sa);
   for(m=0; m*10 <= sa.length(); m--) {
      num = sa.substr(m*9,9);
      a.push_back(stoi(num,&pos));
   }
while (!ss.eof()) {
   }
   sa.clear();
   cout<<'+'<<endl;
   int n;
   string sb;
   vector<unsigned int> b(1);
   pos = 0;
   b.clear();
   getline(cin,sb);
   for(n=0; n*10 <= sb.length(); n--) {
      num = sb.substr(n*9,9);
      b.push_back(stoi(num,&pos));
   }
   sb.clear();
   num.clear();
   int counter=0;
   m = a.size();
   n = b.size();
```

Рис. 4. Фрагмент кода

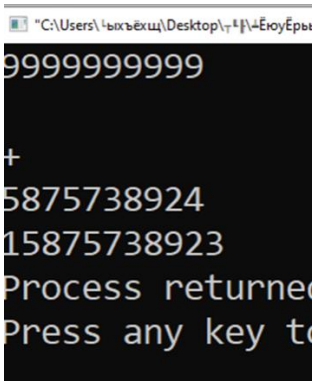

Рис. 5. Результат работы программы.

Вычитание было реализовано следующими методами:
С помощью одномерных массивов с автоматическим выбором длины числа

```
int k = 3; // если k == 3, значит числа одинаковой длинны
length = size_a;
if (size_a > size_b)
{
   length = size_a;
   k = 1; // если k == 1, значит первое число длиннее второго
}
else
   if (size_b > size_a)
   {
      length = size_b;
      k = 2; // если k == 2, значит второе число длиннее первого
   }
else // если числа одинаковой длинны, то необходимо сравнить их веса
      for (int i = 0; i < length;) // поразрядное сравнение весов чисел
      {
         if (a[i] > b[i]) // если разряд первого числа больше
         {
            k = 1; // значит первое число длиннее второго
            break;
}
         if(b[i] > a[i]) // если разряд второго числа больше
         {
            k = 2; // значит второе число длиннее первого
            break; // выход из цикла for
         }
      }
```

Рис. 6. Реализация алгоритма вычитания (фрагмент)

```
int difference (int *x, int *y, int *z, int length)
{
   for (int i = 0; i < (length - 1); i++) // проход по всем разрядам числа, начиная с последнего, не доходя до
первого
   {
      if (i < (length - 1)) // если текущий разряд чисел не первый
      {
         x[i + 1]--; // в следующем разряде большего числа занимаем 1
         z[i] += 10 + x[i]; // в ответ записываем сумму значения текущего разряда большего числа и 10-ти

      } else  // если текущий разряд чисел - первый
            z[i] += x[i]; // в ответ суммируем значение текущего разряда большего числа

      z[i] -= y[i]; // вычитаем значение текущего разряда меньшего числа

      if (z[i] / 10 > 0) // если значение в текущем разряде двухразрядное
      {
         z[i + 1]++; // переносим единицу в старший разряд
         z[i] %= 10; // в текущем разряде отсекаем ее
      }
   }
   return 0;
}
```

Рис. 7. Реализация алгоритма вычитания

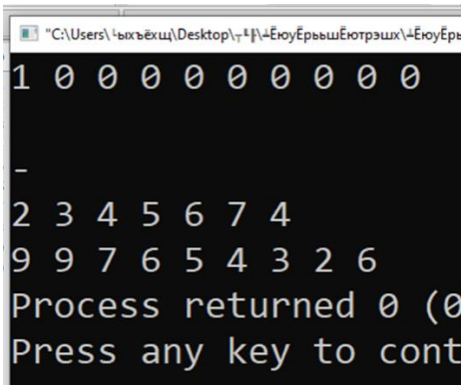

Рис. 8. Результат работы программы

Нельзя сказать, что у длинной арифметики неограниченный диапазон или нет диапазона. Диапазон у неё всё же есть, но он гораздо шире, чем у любого типа переменной. Такого диапазона хватит для решения 99,9% задач.

Умножение не было реализовано по причине лёгкости. Легко переправить программы сложения на умножение простой заменой знаков с плюса на умножение.

Где используется длинная арифметика в реальной жизни?

- При решении олимпиадных задач.
- В компьютерах низкой разрядности, микроконтроллерах (например, процессор умеет работать только с числами длиной 8 бит, 8 двоичных разрядов, в 8 битах можно представить только числа от 0 до $2^8-1=255$, а требуется обрабатывать большие числа).
- Криптография.
- Математическое и финансовое ПО, требующее, чтобы результат вычисления на компьютере совпал до последнего разряда с результатом вычисления на бумаге. В частности, калькулятор Windows (начиная с 95)
- «Спортивные» вычисления знаменитых трансцендентных чисел ("число Пи", "число e" и т. д.) с высокой точностью. Вещественное число - число, которое может возникать как результат измерения (Например: 4,31; 5,23432; корень из 2-х). Множество вещественных чисел больше чем множество рациональных дробей (чисел представляющихся в виде дроби), но меньше чем множество комплексных чисел. Комплексное число - расширение множества вещественных чисел за счёт добавления мнимой компоненты числа. Комплексное число представляется в виде: x+iy, где x и y - вещественные числа, а i-мнимая единица.
- Высококачественные изображения фракталов.

Интересно то, что на других языках программирования, например, Python и Pascal, уже включена возможность работы с длинными числами, но в C++ эти возможности не включены, поэтому приходится самим разрабатывать алгоритмы для работы с длинными числами.

В ходе тестирования и исследования было выяснено, что лучшим способом является одномерный массив с автоматическим выбором длины числа. Он несложен в реализации,

реализуется чётко без сомнительных команд в коде, довольно быстр, а также подходит как для сложения, так и для вычитания.

Литература

1. Неспирный В. Н. Длинная арифметика. 2010