Лекция 5б. Язык С. Массивы и списки.

1. Массивы и списки

Массивы – векторы, матрицы, тензоры.

Элементами массива могут быть базовые типы данных (целые, вещественные, символьные, логические), структуры и объединения, а также адреса этих данных.

Пример программы.

```
>cat massiv1.c
#include <stdio.h>
#include <stdlib.h>
int main ()
{
// Указатели на массивы
  double **A; // Матрица системы Ax=b
  double *B; // Вектор правой части
  double *X; // Вектор неизвестных
  double *C; // Матрица системы в развернутом в строку виде
  int i, j, ij, n;
  char ch = 0;
// Начало алгоритма:
  Met:
  printf("Input massiv dimension ");
  scanf ("%d",&n);
  printf("n=%d\n", n);
// Выделение памяти для массивов
  A = (double**)(malloc(sizeof(double*)*n));
  for (i=0; i<n; i++)
    A[i] = (double*)(calloc(n, sizeof(double)));
  B = (double*)(malloc(sizeof(double)*n));
  X = (double*)(calloc(n, sizeof(double)));
  C = (double*)(calloc(n*n, sizeof(double)));
// Заполнение элементов матрицы
  for (i=0; i<n; i++) {
    for (j=0; j<n; j++) {
      if (j==i) A[i][j] = 1.0*(i+1);
      else      A[i][j] = -0.1/(i+j);
      ij = i*n + j;
      C[ij] = A[i][j];
    }
  }
// Заполнение элементов вектора правой части
  for (i = 0; i < n; i ++) {
    double el;
    printf("Input element with number %d = ", i);
    scanf("%lf",&el);
    B[i] = el;
  }
// Вывод элементов матрицы
  for (i=0; i<n; i++) {
    for (j=0; j<n; j++) {
      printf ("A[%d][%d]=%lf\n", i, j, A[i][j]);
    }
  }
// Вывод элементов вектора правой части
  for (i = 0; i < n; i ++) {
    printf ("B[%d]=%lf\n", i, B[i]);
  }
// Решение системы:
...
// Выдача результатов:
  for (i = 0; i < n; i ++) {
    printf ("X[%d]=%lf\n", i, X[i]);
  }
// Освобождение памяти, занятой массивами:
  for (i=0; i<n; i++) free(A[i]);
  free(A); free(B); free(X); free(C);
```

```
// Продолжение расчетов или выход из программы:
  printf("\n");
  ch = getchar();
  printf("Continue?(y|n): ");
  ch = getchar();
  printf("\n");
  if (ch == 'y') {
    goto Met;
  }

  return 0;
}
```

2. Использование массивов. Метод Гаусса

Задача

Пусть исходная система выглядит следующим образом:

$$\begin{cases} a_{11}x_1 + \ldots + a_{1n}x_n = b_1 \\ \ldots \\ a_{m1}x_1 + \ldots + a_{mn}x_n = b_m \end{cases}$$

Её можно записать в матричном виде:

$$Ax = b,$$

где

$$A = \begin{pmatrix} a_{11} & \ldots & a_{1n} \\ \ldots & & \\ a_{m1} & \ldots & a_{mn} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}. \quad (1)$$

Матрица A называется основной матрицей системы, b — столбцом свободных членов.

Необходимо найти решение – вектор **x**.

Идея Гаусса – с помощью эквивалентных линейных преобразований привести матрицу системы к верхнему треугольному виду:

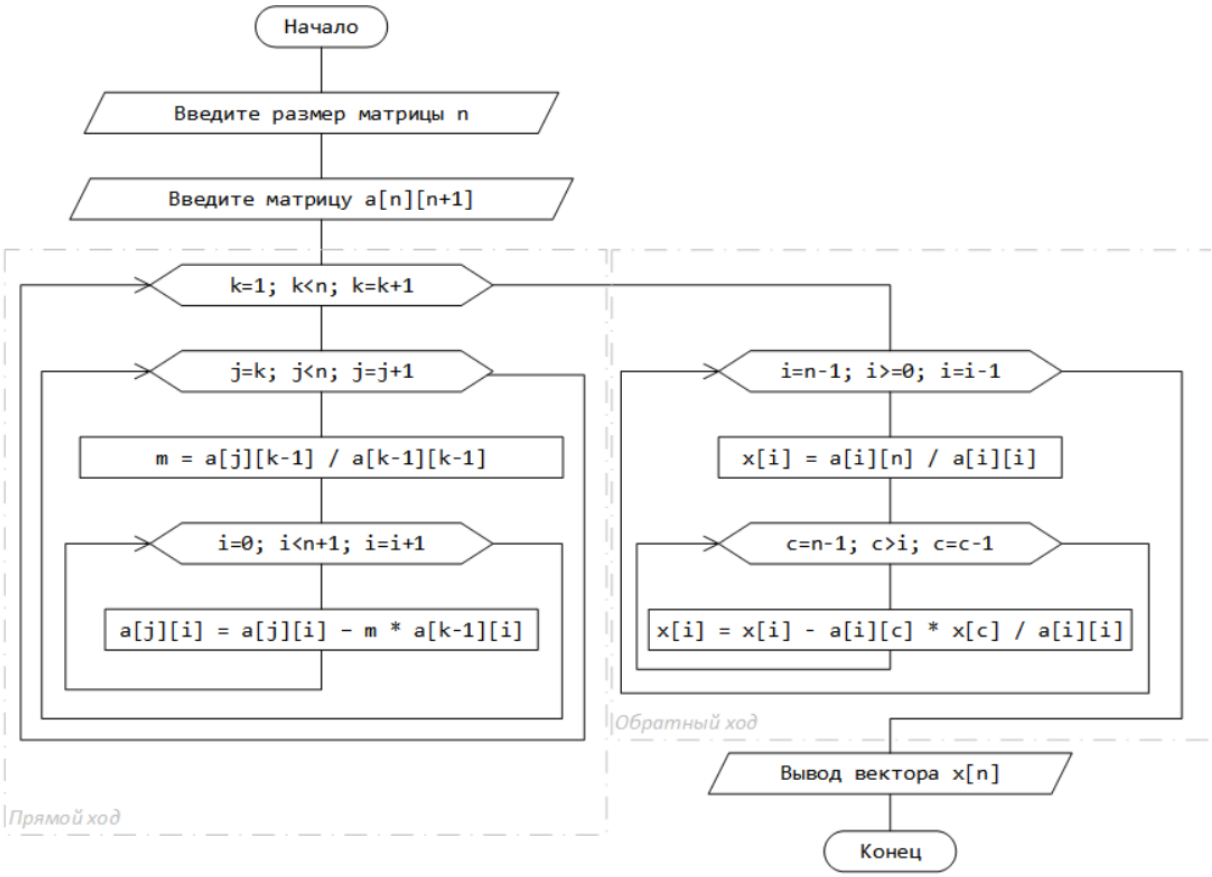
$$\begin{cases} \alpha_{1j_1}x_{j_1} + \alpha_{1j_2}x_{j_2} + \ldots + \alpha_{1j_r}x_{j_r} + \ldots + \alpha_{1j_n}x_{j_n} = \beta_1 \\ \alpha_{2j_2}x_{j_2} + \ldots + \alpha_{2j_r}x_{j_r} + \ldots + \alpha_{2j_n}x_{j_n} = \beta_2 \\ \ldots \\ \alpha_{rj_r}x_{j_r} + \ldots + \alpha_{rj_n}x_{j_n} = \beta_r \\ 0 = \beta_{r+1} \\ \ldots \\ 0 = \beta_m \end{cases}$$

Алгоритм Гаусса. Алгоритм решения СЛАУ методом Гаусса подразделяется на два этапа.

На первом этапе осуществляется так называемый **прямой ход**, когда путём элементарных преобразований над строками систему приводят к треугольной форме, либо устанавливают, что система несовместна. Для этого среди элементов первого столбца матрицы выбирают ненулевой, перемещают содержащую его строку в крайнее верхнее положение, делая эту строку первой. Далее ненулевые элементы первого столбца всех нижележащих строк обнуляются путём вычитания из каждой строки первой строки, домноженной на отношение первого элемента этих строк к первому элементу первой строки. После того, как указанные преобразования были совершены, первую строку и первый столбец мысленно вычёркивают и продолжают, пока не останется матрица нулевого размера. Если на какой-то из итераций среди элементов первого столбца не нашёлся ненулевой, то переходят к следующему столбцу и проделывают аналогичную операцию.

На втором этапе осуществляется так называемый **обратный ход**, суть которого заключается в том, чтобы выразить все получившиеся базисные переменные через небазисные и построить фундаментальную систему решений, либо, если все переменные являются базисными, то выразить в

численном виде единственное решение системы линейных уравнений. Эта процедура начинается с последнего уравнения, из которого выражают соответствующую базисную переменную (а она там всего одна) и подставляют в предыдущие уравнения, и так далее, поднимаясь по <ступенькам> наверх. Каждой строчке соответствует ровно одна базисная переменная, поэтому на каждом шаге, кроме последнего (самого верхнего), ситуация в точности повторяет случай последней строки.



Блок-схема алгоритма Гаусса.

Пример программы

```
>cat massiv2.c
#include <stdio.h>
#include <stdlib.h>
void print_system(int n, double **A, double *X);
int main ()
{
// Указатели на массивы
  double **A; // Матрица системы
  double *B; // Вектор правой части
  double *X; // Вектор неизвестных
  double *C; // Матрица системы в развернутом в строку виде
  int i, j, k, ij, n, ier;
  double s, p;
  char ch = 0;
// Начало алгоритма:
  Met:
  printf("Input massiv dimension ");
  scanf ("%d",&n);
  printf("n=%d\n", n);
// Выделение памяти для массивов
  A = (double**)(malloc(sizeof(double*)*n));
  for (i=0; i<n; i++)
    A[i] = (double*)(calloc(sizeof(double),n));
  B = (double*)(malloc(sizeof(double)*n));
  X = (double*)(calloc(sizeof(double),n));
  C = (double*)(calloc(sizeof(double*),n*n));
// Заполнение элементов матрицы
  for (i=0; i<n; i++) {
    for (j=0; j<n; j++) {
```

```
      if (j==i) A[i][j] = 1.0*(i+1);
      else      A[i][j] = -0.1/(i+j);
      ij = i*n + j;
      C[ij] = A[i][j];
    }
  }
// Заполнение элементов вектора правой части
  for (i = 0; i < n; i ++) {
    double el;
    printf("Input element with number %d = ", i);
    scanf("%lf",&el);
    B[i] = el;
  }
// Вывод элементов матрицы
  for (i=0; i<n; i++) {
    for (j=0; j<n; j++) {
      printf ("A[%d][%d]=%lf\n", i, j, A[i][j]);
    }
  }
// Вывод элементов вектора правой части
  for (i = 0; i < n; i ++) {
    printf ("B[%d]=%lf\n", i, B[i]);
  }
  printf("\n");
// Решение системы:
  ier =0;
  for (i=0; i<n; i++) X[i] = B[i];
  print_system(n,A,X);

  for (i=0; i<n; i++) {
    s = A[i][i];
    if (s!=0) {
      s = 1.0/s;
      A[i][i] = 1.0;
      for (j=i+1; j<n; j++) A[i][j] *= s;
      X[i] *= s;
      if (i<n-1){
        for (k=i+1; k<n; k++){
          p = A[k][i];
          A[k][i] = 0;
          for (j=i+1; j<n; j++) A[k][j] -= (p*A[i][j]);
          X[k] -= (p*X[i]);
        }
      }
      print_system(n,A,X);
    }
    else {
      ier = -1;
      printf("Bad matrix!\n");
      break;
    }
  }
  if (ier == 0){
    for (i=(n-2); i>=0; i--) {
      s = 0;
      for (j=i+1; j<n; j++) s += (A[i][j]*X[j]);
      X[i] -= s;
    }
    print_system(n,A,X);
  }
// Проверка решения:
  if (ier == 0){
    for (i=0; i<n; i++) {
      s = B[i];
      for (j=0; j<n; j++) {
        ij = i*n + j;
        s -= (C[ij]*X[j]);
      }
```

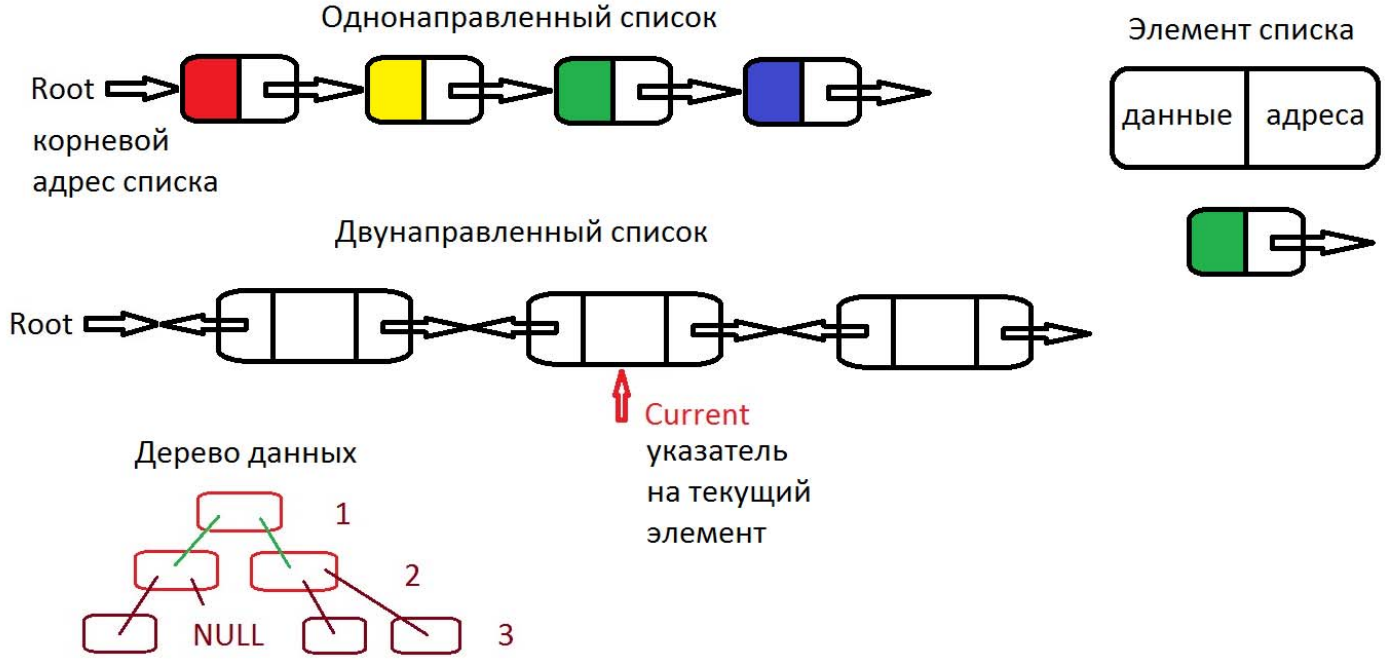
```
      printf("i=%d s=%le\n",i,s);
    }
  }
// Освобождение памяти:
  for (i=0; i<n; i++) free(A[i]);
  free(A); free(B); free(X); free(C);
// Продолжение расчетов или выход из программы:
  printf("\n");
  ch = getchar();
  printf("Continue?(y|n): ");
  ch = getchar();
  printf("\n");
  if (ch == 'y') {
    goto Met;
  }
  return 0;
}

void print_system(int n, double **A, double *X)
{
  int i, j;
  for (i=0; i<n; i++){
    for (j=0; j<n; j++) printf("%14.6le ",A[i][j]);
    printf("%14.6le\n",X[i]);
  }
  printf("\n");
  return;
}
```

3. Списки и деревья

Список – набор данных, элементами которого являются данные и их адреса.

Дерево отличается от списка наличием нелинейной структуры связей (графа связей).



Пример работы со списками:

```
>cat lists.c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <math.h>

typedef struct Abonent_t {
  char FirstName[20];
  char LastName[20];
```

```
  char PhoneCode[4];
  char PhoneNumber[11];
  struct Abonent_t* NextLeft;
  struct Abonent_t* NextRight;
} Abonent_t;

typedef struct Book_t {
  int Length;
  struct Abonent_t* First;
} Book_t;

void PrintBook(Book_t Book)
{
  int i;
  struct Abonent_t* Ab;
  struct Abonent_t  Abb;
  struct Abonent_t* Abbb = &Abb;

Ab->Next = NULL;

Abb.Next = NULL;
Abbb->Next = NULL;

  printf("Book length is %d\n", Book.Length);

  if (Book.Length > 0) {
    for (i=0; i<Book.Length; i++){
      if (i==0) Ab = Book.First;
      else      Ab = Ab->Next;
      printf("Abonent %d is %s %s %s %s\n",
        (i+1), Ab->FirstName, Ab->LastName, Ab->PhoneCode, Ab->PhoneNumber);
    }
  }

  return;
}

int main() {
  int i, n;

  Book_t Book;
  Abonent_t *Ab, *Ac;

  double ddd, fff;
  double *eee;
  double *aaa, *bbb;

eee = &ddd;
*eee = 12.34;

eee = &fff;
*eee = 35.67;

  Ab = (Abonent_t*)calloc(1, sizeof(Abonent_t));
  strncpy(Ab->FirstName,"Ivan",4);
  strncpy(Ab->LastName,"Petrov",6);
  strncpy(Ab->PhoneCode,"007",3);
  strncpy(Ab->PhoneNumber,"4953222231",10);
  Ab->Next = NULL;

  Book.Length = 1;
  Book.First  = Ab;

  PrintBook(Book);

  Ab->Next = (Abonent_t*)calloc(1, sizeof(Abonent_t));
  Ac = Ab->Next;
  strncpy(Ac->FirstName,"Nikolay",7);
```

```
  strncpy(Ac->LastName,"Sidorov",7);
  strncpy(Ac->PhoneCode,"007",3);
  strncpy(Ac->PhoneNumber,"4998445416",10);
  Ac->Next = NULL;

  Book.Length = 2;
  PrintBook(Book);

  Ac = (Abonent_t*)calloc(1, sizeof(Abonent_t));
  strncpy(Ac->FirstName,"Petr",4);
  strncpy(Ac->LastName,"Grigoriev",9);
  strncpy(Ac->PhoneCode,"001",3);
  strncpy(Ac->PhoneNumber,"3458995416",10);
  Ac->Next = NULL;

  Ab = Book.First; // first element
  Ab = Ab->Next;   // second element
  Ab->Next = Ac;   // link to third element

  Book.Length = 3;
  PrintBook(Book);

  Ac = NULL;
  for (i=0; i<Book.Length; i++){
    if (i==0) Ab = Book.First;
    else      Ab = Ab->Next;

    if ((i+1) == 2) {
      Ac->Next = Ab->Next;
      free(Ab);
      Ab = NULL;
      Book.Length = 2;
      break;
    }

    Ac = Ab; // Previous element
  }

  PrintBook(Book);

  return 0;
}
```