Лекция 4. Язык С. Основные конструкции языка.

1. Операции, операторы и конструкции.

1.1. Операции.

В языке С используются следующие элементарные операции:

Операция	Наименование	Операция	Наименование
!	Логическое НЕ	^	Поразрядное исключающее ИЛИ
~	Обратный код	&&	Логическое И
+	Сложение; унарный плюс	||	Логическое ИЛИ
-	Вычитание; унарный минус	?:	Условная операция
*	Умножение; косвенная адресация	++	Инкремент
/	Деление	--	Декремент
%	Остаток от деления	=	Простое присваивание
<<	Сдвиг влево	+=	Присваивание со сложением
>>	Сдвиг вправо	-=	Присваивание с вычитанием
<	Меньше	*=	Присваивание с умножением
<=	Меньше или равно	/=	Присваивание с делением
>	Больше	%=	Присваивание с остатком от деления
>=	Больше или равно	>>=	Присваивание со сдвигом вправо
==	Равно	<<=	Присваивание со сдвигом влево
!=	Не равно	&=	Присваивание с поразрядным И
&	Поразрядное И; адресация	|=	Присваивание с поразрядным включающим ИЛИ
|	Поразрядное включающее ИЛИ	^=	Присваивание с поразрядным исключающим ИЛИ
,	Последовательное выполнение (запятая)		

1.2. Операторы и конструкции.

Операторы:

1) пустой оператор (разделитель между операторами)

```
;
```

Пример:

```
a=1.25;
```

2) составной оператор или блок

```
{ <объявления> <операторы> }
```

Пример:

```
{ double x=1.0, y, z; y=x+2.0; z=x*y; }
```

3) оператор-выражение

Примеры:

```
x = x + 1.0; x += 1.0; x++1.0; x++; x++3.0;
x = y+3; z=f(x); y = sin(x); t=2.0*sin(x)*cos(x); v=sin(2.0*x);
```

4) условный оператор if

```
if(<выражение>) <оператор1> [else <оператор2>]
```

Примеры:

```
if (x>0) y = 1;
if (x<0) y = 2; else y = 0;

if (x>0) y = 1; else if (x<0) y = 2; else y = 0;

if (x>0) { y = 1; x = 2; }
if (x<0) y = 2; else y = 0;

if (x>0) { y = 1; x = 2; } else if (x<0) y = 2; else y = 0;

double a=1.0, b=2.0, c=0.5;
int i = 0;
// c < a, c < b => i = 1
// c = a, c < b => i = 2
// c > a, c < b => i = 3
// c > a, c = b => i = 4
```

```
// c > a, c > b => i = 4

if (c > a) { if (c < b) i = 3; else if (c == b) i=4; else i = 5;}
else { if (c == a) i = 2; else i = 1;}
if (c=a) i = 2; !!! – присваивание c=a;
```

5) оператор пошагового цикла for

```
for([<начальное-выражение>]; [<условное-выражение>]; [<выражение-приращения>]) <оператор>
```

Примеры:

```
int i, n=10;
double x, y;

for (i=0; i<=n; i++;)  { x=(Pi/n)*i; y=sin(x); printf("x=%le y=%le\n",x,y);}
for (i=1; i<=n; i**2;) { x=(Pi/n)*i; y=sin(x); printf("x=%le y=%le\n",x,y);}
```

6) оператор цикла с предусловием while

```
while (<выражение>) <оператор>
```

Примеры:

```
int x=1, y=0, z=10;
while (x == 1) { y += 2; if (y == 8) x = 3; }
x = 7;
while (x != 1) { y -= 3; z += y; if (z < 0) x = 1; }
```

Оператор while может привести к зацикливанию программы !!!

7) оператор цикла с постусловием do

```
do <оператор> while (<выражение>);
```

Примеры:

```
int x=1, y=0, z=10;
do { y += 2; if (y == 8) x = 3; } while (x == 1);
x = 7;
do { y -= 3; z += y; if (z < 0) x = 1; } while (x != 1);
```

Оператор do может привести к зацикливанию программы !!!

8) оператор продолжения continue

```
continue;
```

9) оператор-переключатель switch

```
switch(<выражение>)
{
  [<объявление>]
  [case <константное-выражение>:] [<оператор>]
  [case <константное-выражение>:] [<оператор>]
  [default:] [<оператор>]
}
```

Пример:

```
if (ch == "A") ch="a";
else {
   if (ch == "B") ch="b";
   else {
      if (ch == "C") ch="c";
      ...
   }
}
...
switch(ch)
{
   case 'A': ch="a";
   case 'B': ch="b";
   case 'C': ch="c";
   ...
   default: printf("We have non-printable symbol!\n");
}
```

Использование switch для вещественных чисел ограничено ошибками округления!!!

10) оператор разрыва break (выход из циклов for, while, do)

```
break;
Пример:
while (1 == 1) { // бесконечный цикл
…
  if (x > y) break; // выход из цикла
}
```

11) оператор перехода goto (переход на метку)

```
goto <метка>;
...
<метка>: <оператор>
Пример:
if(x<0) goto Met1; else if(x>0) goto Met2;
<действия для x=0> goto Met3;
Met1: <действия для x<0> goto Met3;
Met2: <действия для x>0>
Met3: <конец>
```

12) оператор возврата return (выход из функции)

```
return [<выражение>];
Пример:
Met3: return;
Met3: return 0;
```

1.3. Арифметика и логика.

Компьютер может выполнять только арифметические и логические операторы. Поэтому все другие действия, являются результатом выполнения арифметических и логических операций.

Действия с целыми числами: - элементарные арифметические операции; - элементарные логические операции; - операции побитовой обработки; - операции сравнения; - вычисление функций.	Действия с символами: - элементарные арифметические операции; - элементарные логические операции; - операции побитовой обработки; - операции сравнения; - вычисление функций; - формирование строк.
Действия с вещественными числами: - элементарные арифметические операции; - операции сравнения; - вычисление функций.	Действия с логическими переменными: - элементарные логические операции; - операции побитовой обработки; - операции сравнения; - вычисление функций.

2. Массивы.

Массив – это упорядоченный набор элементов одинакового типа. Массив может содержать элементы базовых типов (целые или вещественные числа, символы, логические переменные) или пользовательских типов (адреса, структуры, объединения).

С математической точки зрения массив – вектор, матрица, тензор, многомерная таблица.

У каждого массива имеется два типа параметров – число измерений и длина по каждому измерению.

Массивы бывают статическими (измерения и длины которых указываются явно при описании) и динамическими (при описании для них указывается лишь число измерений).

При описании статических массивов используется конструкция:

<тип> <имя>[n1]...[n2];

```
Примеры:
int a[5];          // целочисленный вектор длины 5
double b[3][6];    // вещественная матрица с длинами 3 и 6 (3x6)
float c[4][4][4]; // тензор 3го ранга с длинами 4 (4x4x4)
char s[10];        // строка из 10 символов
MySruct St[20];    // массив структур, содержащий 20 элементов
```

При описании динамических массивов используется конструкция:

<тип> <*...*><имя>

Количество '*' соответствует размерности массива

Примеры:
```
int *a;       // адрес целочисленного вектора неизвестной пока длины
double **b;   // адрес вещественной матрицы с неизвестными пока длинами строк и столбцов
float ***c;   // адрес тензора 3го ранга с неизвестной пока длиной по всем измерениям
char *s;      // адрес строки с неизвестным пока количеством символов
MySruct *St;  // мадрес массива структур с неизвестным пока количеством элементов
```
Статические массивы размещаются в памяти (в стеке программы) сразу после старта программы одним блоком, даже если они многомерные.

Примеры:
```
int a[5];       // целочисленный вектор занимает в памяти стека блок из 5*4 = 20 байт
double b[3][6]; // вещественная матрица занимает в памяти стека блок из 3*6*8 = 144 байт
```
Динамические массивы выделяются только по необходимости с помощью функций malloc или calloc и размещается в динамической памяти ("куче") несколькими блоками: сначала адреса, затем данные.

Примеры:
```
int *a; double **b; // в стеке определены адреса будущих массивов,
                    // размеры адресов - 8 байт каждый
...
a = (int *)malloc(sizeof(int)*10); // с этого момента занимает дополнительно
                                   // в памяти кучи блок из 10*4 = 40 байт
...
b = (double **)malloc(sizeof(double *)*10); // с этого момента занимает дополнительно
                                            // в памяти кучи блок из 10*8 = 80 байт
                                            // в этом блоке хранятся адреса строк
for (i=0; i<10; i++) b[i] = (double *)malloc(sizeof(double)*10);
// с этого моменты выделены 10 блоков по 10*8 = 80 байт для каждой строки матрицы
```

3. Функции.

3.1 Понятие функции

В математике y=f(x) – отображение одного пространства на другое в числовом выражении.
Функция может быть скалярной или векторной.
В языке С функции являются скалярными и имеют строго определенный тип. Аргументы функций строго типизированы.

Примеры:
```
void MyFun(); => return;
double MyFun(); => return 3.14;
```

3.2 Библиотеки функций

Основная библиотека (stdlib) содержит множество функций общего назначения.
Ее подмножество – математические функции: exp, log, pow, sin, cos, tan, atan, ...

3.3 Программирование новых функций

В языке С имеется возможность создавать новые функции.
Пример:
```
double MyFun1(double x); // объявление функции 1
double MyFun2(double); // объявление функции 2
...
int main()
{
  ...
  return 0;
}

// реализация функции 1
double MyFun1(double x)
{
  return ((x-1.0)/(x-1.0));
}
// реализация функции 2
double MyFun2(double t)
{
  double y;
  y = sin(t)+1.0;
  return y;
}
```