Лекция 2. Обзор языка C, продолжение

Как знакомиться с новым алгоритмическим языком?

0. идеология языка, что делает программа и как она строится;
1. базовые типы данных и переменные;
2. операции с базовыми типами, выражения;
3. формирование производных типов данных;
4. операторы и управляющие конструкции;
5. структуризация записи программы (блоки, объявления, процедуры , контексты и т.п.);
6. видимость объектов (локальные, глобальные, статические, автоматические и т.д.);
7. организация ввода-вывода, внешние и стандартные библиотеки;
8. прочие специфические особенности и возможности;
9. запуск и отладка программы в конкретной системе программирования.

3. Формирование производных типов данных.

Массивы уже обсудили.

Есть еще структуры, объединения, битовые поля — о них будет отдельный разговор.

4. Операторы и управляющие конструкции.

Оператор присваивания:

```
lvalue = выражение ;
a[2*k+1] = a[2*k-1] + 3.14*r;
b = c = d = 1;                  a = (b=3) + 2;      b = 3;
                                                    a = 5;
```

Разветвления:

```
if (выражение)          if (выражение)        можно вкладывать
{                       {                     друг в друга
   операторы-true          операторы-true
}  else {               }
   операторы-false
}
```

```
if (x < 3)
{
   y = 1;
} else {
   z = 2*x + 5;
}
```

```
if ( x >= 0)
{
   y = x;
} else {
   y = -x;
}
```

```
ловушка для новичка:
if (x==1) { .... }
if (x=1) { .... }
```

Условное выражение:

```
(выражение_0) ? выражение_1 : выражение_2
y = (x>=0) ? x : -x;
```

Переключатель:

```
switch (целое выражение)
{
   case константа1 :
      операторы
      break;
   case константа2 :
      операторы
      break;
   ...

   ...
   ...
   default:
      операторы
}
```

```
switch (k)
{
   case 0:
     y = 0;
     break;
   case 25:
     y = 1;
     break;
   case 100:
     z = 333;
   case 200:
   case 300:
     y = 2;
     break;
   default:
     y = -1;
}
```

Циклы:

```
while ( выражение )
{
    тело цикла
}
```

```
int a[10], i = 0;
while (i < 10)
{
    a[i++] = 0;
}
```

```
while( true)
{

}
```

```
do
{
    тело цикла
} while (выражение);
```

```
int a[10], i = 10;
do
{
   a[--i] = 0;
} while(i);
```

Цикл for:

```
                                          эквивалентно
for (выражение1; выражение2; выражение3)         выражение1;
 {                                               while(выражение2)
     тело цикла                                  {
 }                                                   тело цикла
                                                     выражение3;
                                                 }
```

```
for (i=0; i<n; i++)
for (i=n; i>0; i--)
for (i=1; i<n; i*=2)
for (i=1, j=n; i<j; i++, j--)
```

```
break
continue
```

```
        for (i=0; i<n; i++)            k = 0;
        {                              while(k<100)
            a = ....                   {
            if (a<0) continue;             a = .....
            b = ....                       if (a < 0) break;
        }                                  k++;
                                       }
```

5. Структуризация записи программы (блоки, объявления, процедуры , контексты и т.п.)

Основные понятия:

```
блок    {...}
{
   .....
}
```

Функция:

```
тип возвр.значения  имя (список параметров)      // это заголовок функции
{
   тело функции
}

double fun(int x, double y)    // определение функции
{
```

```
    double z;                   // локальные переменные
    z = x + y;                  // содержательные операции
    return 2*z;                 // возврат значения
}
....
int a,b,c;
double k,l,m;
....
k = fun(a, m);
l = fun(a, m) + 3*fun(b,c);
```

Объявления переменных:

```
int x;
double y = 0;
```

Объявления функций (проотип):

```
double fun(int x, double y);     // заголовок функции
int g(int, int, char);           // можно без имен аргументов

void    - "отсутствие"
void  AnotheFunction (int x);
int   OneMoreFun(void);
```

Программа — последовательность объявлений функций и пременных и определений функций.

Объявления задают области видимости переменных и функций.

Начальная функция (точка входа) — функция main — с нее начинается выполнение программы

```
int main(void)          // есть и другие варианты заголовка ...
```

Программа может быть записана в нескольких файлах.

Блок {} задает контекст блока.

Файл задает контекст файла.

Объявления и определения действуют в пределах контекста.

Мы сразу будем рассматривать программу в нескольких файлах (в учебных целях, хотя иногда без этого можно обойтись).

6. Видимость объектов (локальные, глобальные, статические, автоматические и т.д.).

Объявления задают область видимости переменной или функции в пределах контекста всюду после этого объявления.

```
{          // блок                    ........ файл
  int a;                              .  int b;
  ....                                .  ....
}                                     .  ....
                                      .
                                      ........ конец файла
```

```
   file1              file2            file3            file4
<объявление>       <объявление>     <объявление>     <объявление>
<функция>          <объявление>     <объявление>     <функция>
<объявление>       <объявление>     <объявление>     <функция>
<функция>          <функция>        <объявление>     <функция>
<функция>          <функция>
<функция>
```

Переменные и функции определяются в контесте файла только а одном месте (в одном файле).

```
   file1              file2
   int a;             int a;
так нельзя - множественное определение
```

Чтобы расширить область видимости переменной или функции в другой файл, там следует записать их объявление со словом extern. Для объявления функций слово extern можно не писать.

```
   file1              file2            file3
   int a;          extern int a;    int f(int x) {...}
int f(int);        здесь f не видно  здесь f видно
здесь a видно      здесь a видно     здесь a не видно
здесь f видно
```

Ключевое слово static при определении переменной или функции в контексте файла ограничивает область видимости только этим файлом.

```
   file1              file2
static int a;      static int a;
здесь a свое       здесь a свое
и не смешивается с a из другого файла
```

Ключевое слово static при определении переменной в контексте блока сохраняет эту переменную при выходе из блока. Вне блока эта переменная не видна, но при повторном входе в блок ей можно продолжать пользоваться, и она сохраняет свои значения с предыдущего посещения блока.