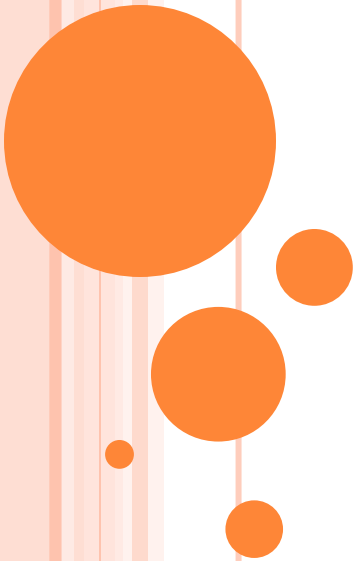


Функции

Лекция №9

Подходы к разработке ПО

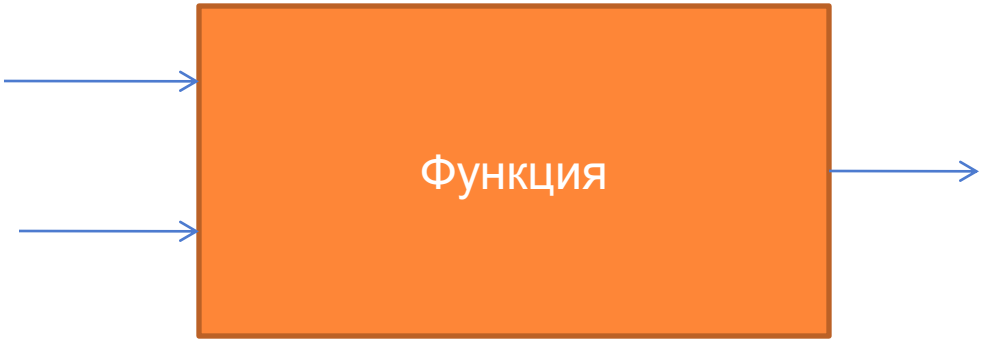
- Метод «северо-западного» угла
  - Программа пишется как единое целое
  - В сложной программе трудно найти и исправить ошибку
  - Из-за небольших изменений требований приходится все переделывать
- Модульное программирование
  - Возможность вызова одного модуля для разных исходных данных
  - Возможность независимой отладки отдельных модулей

Исходные предпосылки

- Функция – основная структурная единица программы на Си
- Действия вне функции не выполняются
- Функция обычно имеет n входов и 1 выход
- У функции могут отсутствовать входные или выходные параметры (void)
- Для решения конкретных задач стандартных функций не хватает – приходится писать самому

Функция

- Логически завершенный программный модуль, который решает конкретную небольшую задачу



Состав функции

- Заголовок
  - Тип результата
  - Имя функции
  - Описание входных параметров
- Тело
  - Описание локальных переменных
  - Операторы
  - Оператор возврата из функции return

Формат описания функции

тип_рез имя_ф_ии (список_формальных_парам)
{
описания локальных переменных;
операторы;
}

- Для каждого формального параметра указывается тип данных
- Формальные параметры разделяются запятой
- При отсутствии формальных параметров вместо них указывается void
- При отсутствии результата вместо типа результата - void

Имя функции

- Должно быть уникальным
- Состоит из латинских букв и цифр
- Начинается с буквы

Список формальных параметров

- Формальные параметры – параметры, которым реальные значения присваиваются при вызове функции
  - перечисляются через запятую
  - для каждого параметра указываются имя и тип
  - при вызове функции вместо формальных параметров подставляются фактические параметры
  - формальные параметры видимы только внутри тела функции
- При отсутствии формальных параметров вместо их списка указывается слово void

Тип результата

- Тип выражения, значение которого возвращается в точку вызова с помощью команды return
- При отсутствии возвращаемого результата вместо типа результата указывается слово void

Локальные переменные

- Переменные, описываемые в начале тела функции
  - видимы только внутри функции
  - могут использоваться для промежуточных преобразований формальных параметров и формирования результата работы функции

Операторы

- Любые операторы языка Си
- Вызовы других функций
- Оператор возврата результата return

- Формат оператора return
  - при наличии результата
    - return(выражение);
  - при отсутствии результата
    - return;

Пример описания функции

```
//выбор наибольшего числа
int fMaxNum(int a, int b)
{int c;
if(a>b)
   c=a;
else
   c=b;
return c;
}
```

Вызов функции

- Функция вызывается для конкретных фактических параметров, копии которых подставляются вместо формальных параметров
- Фактические параметры
  - Переменные
  - Константы
  - Выражения

- Формат вызова функции
  - имя_функции(список фактических параметров)

Правила вызова функции

- Имя функции при вызове должно совпадать с именем функции при описании
- Количество фактических параметров должно быть равно количеству формальных параметров
- Очередности формальных и фактических параметров должны совпадать
- Типы фактических параметров должны соответствовать типам формальных параметров
- Результат вызова должен быть присвоен какой-нибудь переменной, иначе он теряется

Примеры вызова функции

- int a1=2,b1=3,c1,d1,e1;
- c1=fMaxNum(a1,b1);
- printf(“a1=%d b1=%d c1=%d\n”,a1,b1,c1);
- d1=fMaxNum(a1+b1, 4);
- e1=2*fMaxNum(2,b1)+a1;

Пример программы с вызовом функции

```
#include <stdio.h>
//Вычисление суммы ряда чисел
int fSumInt(int m, int n)
{int s=0,i;
for (i=m;i<n;i++)
   s=s+i;
return s;
}
void main(void)
{
int k1=3,k2=9,sum;
sum=fSumInt(k1,k2);
printf("sum from %d to %d is %d",k1,k2,sum);
}
```

Области видимости функций

- По умолчанию функция видна от места описания до конца файла
  - Вызов должен быть ниже описания
- Если функцию нужно вызвать выше описания – в начале файла должен быть протитип

- Прототип – заголовок функции, заканчивающийся точкой с запятой
  - int fSumInt(int m, int n);

Передача массива в функцию

- Передача постоянного адреса начала массива
  - int func(int x[],int n)
  - Вместо размера – пустые квадратные скобки
  - Размер передается отдельным параметром
- Передача адреса любого элемента массива через указатель
  - int func1(int *p,int n)
- В отличие от переменных массив доступен в напрямую из вызываемой функции

Пример функции обработки массива

```
int fMin(int x[],int n)
   {int i, imin=0;
   for(i=1;i<n;i++)
         if(x[i]<x[min}
               imin=i;
   return imin;
   }
```

Вызов функции обработки массива

- int M[]={2,1,5,4,3},*p,k;
- k=fMin(M,5);

- p=M;
- k=fMin(p,5);

- k=fMin(p+1,4);
- printf(“Min M[%d]=%d”,k,p[k]);

Возврат указателя в качестве результата

```
int* fFindVal(int *p, int n, int v)
   {int *q;
    for(q=p;q<p+n;q++)
          if(*q==v)
                  return q;
      return NULL;}
void main(void)
{int X[]={2,1,4,3,5},*q1;
q1=fFindVal(X,5,4);
if(q1==NULL) printf("No element\n");}
```

Программа сортировки массива

- Функция поиска минимума
- Функция упорядочения по возрастанию
- Главная функция

Указатель на функцию

- int (*fptr)(int *p,int n);
- int m;
- int X[5]={2,1,5,3,4};
- fptr=&fMin;
- m=(*fptr)(X,5);

Массив указателей на функции

- int (*fm[2])(int *p,int n);
- int i,Y[2];
- int X[5]={2,1,5,3,4};
- fm[0]=&fMin;
- fm[0]=&fMax;
- for (i=0;i<2;i++)
- Y[i]=(*fm[i])(X,5);

Область видимости и время жизни переменной в программе из нескольких функций

- Локальная переменная
  - Описывается в теле функции
  - Видима только внутри функции
  - Появляется при входе в функцию, исчезает при возврате из нее
- Глобальная переменная
  - Описывается вне функций
  - Видима из любой функции ниже ее определения
  - Появляется в начале работы программы, и исчезает по окончании работы программы

Область видимости и время жизни переменной в программе из нескольких функций

- Статическая переменная
  - Имеет модификатор static и начальное значение 0
  - Видима только внутри функции
  - Появляется в начале работы программы, исчезает в конце работы программы

```
int f(void)
{static int k;
k++;
return k;
}
```

Выводы

- Разбиение программы на функции упрощает отладку программы
- Функцию можно многократно использовать для разных наборов исходных данных
- Фактические параметры должны соответствовать формальным параметрами
- Результат вызова необходимо сохранить