

Семинар №3

ФАКИ 2016

Бирюков В. А.

November 17, 2016

Объявление, определение и вызов функций

Функции

- Функция – фрагмент программного кода, к которому можно обратиться из другого места программы
- Функция должна быть соответствующим образом объявлена и определена

Объявление функций(прототипы функций)

- Как и переменная, функция также должна объявлена перед использованием
- Определение функции называется прототипом функции
- Пример прототипа:

```
int sum (int a, int b)
```

или:

```
int sum (int , int )
```

Вызов функций

- Примеры вызова функции:

Функция, определяемая пользователем:

```
sum(a, b)
```

Библиотечные функции:

```
sqrt(x)
```

```
printf("%d\n", a)
```

Определение функции

Возвращаемый
тип

Имя функции

Параметры
функции

```
int sum ( int a, int b )
```

```
{
```

```
    int c;
```

Объявление локальной
переменной

```
    c = a + b;
```

```
    return ( c );
```

```
}
```

Оператор return
завершает выполнение функции
возвращает значение c

Функции

int sum(int a, int b); ← Прототип функции

```
int main()
{
    printf("%d\n", sum(5, 4));
    return 0;
}
```

↑ Вызов функции


```
int sum(int a, int b)
{
    return a + b;
}
```

← Определение функции

Функции


```
int sum(int a, int b)
{
    return a + b;
}
```

Прототип и
определение
функции



```
int main()
{
    printf("%d\n", sum(5, 4));
    return 0;
}
```

Вызов функции



Зачем разделять прототип функции и её
определение?

Спецификатор типа void

```
void print_int ( int a )  
{  
    printf("%d\n", a);  
}
```

- Указывает на то, что функция ничего не возвращает

Спецификатор типа void

```
int print_int ( void )  
{  
    printf("%d\n", 42);  
    return 42;  
}
```

- Указывает на то, что функция ничего не принимает

Области видимости переменных

Области видимости переменных

- Область видимости – область программы, в пределах которой имя некоторой переменной продолжает быть связанным с этой переменной и возвращать её значение.
- Глобальная переменная – объявляются вне всех функций и доступны отовсюду
- Локальная переменная – объявляются внутри блока и недоступны вне его

Области видимости переменных

Функция определяет собственную (локальную) область видимости, куда входят:

- 1 Глобальные переменные
- 2 Входные параметры
- 3 Переменные, которые объявляются в теле самой функции

Передача по ссылке и значению

Передача по значению

- В памяти создаётся копия передаваемого значения, которое и используется в функции
- Значение передаваемой переменной не изменяется
- Этот способ передачи используется в языке C

Передача по значению

Что выведет программа?

```
void add5(int a){  
    a = a + 5;  
}
```

```
int main() {  
    int a = 10;  
    add5(a);  
    printf("%d\n", a);  
    return 0;  
}
```


Передача по ссылке

- Используется переменная, передаваемая в функцию
- Если её значение в функции изменяется, то и изменяется значение исходной переменной
- Этого способа передачи нет в C
- Но можно эффективно делать почти то же самое при помощи указателей

Передача адреса переменной

Что выведет программа?

```
void add5(int * a){  
    *a = *a + 5;  
}
```

```
int main() {  
    int a = 10;  
    add5(&a);  
    printf("%d\n", a);  
    return 0;  
}
```

Функция main

Функция main

```
int main ( )  
{  
    <Операторы>  
    return ( 0 );  
}
```

- Точка входа в программу
- Возвращает 0, если программа завершилась нормально
- В простейшем виде не принимает аргументов

Функция main

```
int main ( int argc, char * argv[] )  
{  
    <Операторы>  
    return ( 0 );  
}
```

- argv – параметры, передаваемые в функцию main
- argc – количество этих параметров

Функции

Функция main

```
~/ $ gcc -o prog_name prog.c
```



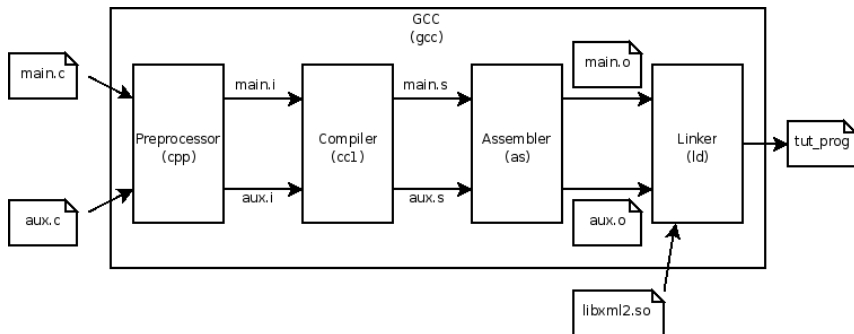
```
argc = 4
```

```
argv = {"gcc", "-o", "prog_name", "prog.c"}
```

Директивы препроцессора `#include` и `#define`

Препроцессор C

- Препроцессор C – программа, подготавливающая код программы к КОМПИЛЯЦИИ



Директива #include

- Подставляет содержимое указанного файла на место этой директивы
- ```
#include <stdio.h>
#include "my_file.txt"
```

# Директива #define – макроподстановка

- Подставляет заданное выражение вместо заданного токена
- ```
#define NUMBER_OF_MONTHS 12  
#define LESS <
```
- Часто используется для задания констант (особенно в старом коде)
- С #define нужно быть аккуратным

Квалификатор типа const

Современное задание констант

```
const int number_of_months = 12;  
const float Pi = 3.14159265;
```

Рекурсия

Рекурсия

- Существует возможность вызвать функцию внутри самой функции
- Такой вызов функции называется рекурсивным

Пример рекурсии: вычисление факториала

```
int fact(int n)
{
    int res;
    if (n <= 1) {
        return 1;
    }
    res = n * fact(n-1);
    return res;
}
```

Задачи

- 1 Все задачи начиная с Vfunc_1 на judge.mipt.ru