

Семинар №3

ФАКИ 2017

Бирюков В. А.

October 9, 2017

Объявление, определение и вызов функций

- 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)
```

- Примеры вызова функции:

Функция, определяемая пользователем:

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

Библиотечные функции:

```
sqrt(x)
```

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

Определение 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


```
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?


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

- Указывает на то, что функция ничего не возвращает

```
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

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

- ❶ Глобальные переменные
- ❷ Входные параметры
- ❸ Переменные, которые объявляются в теле самой функции

Передача по ссылке и 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;  
}
```


- 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;  
}
```

Рекурсия

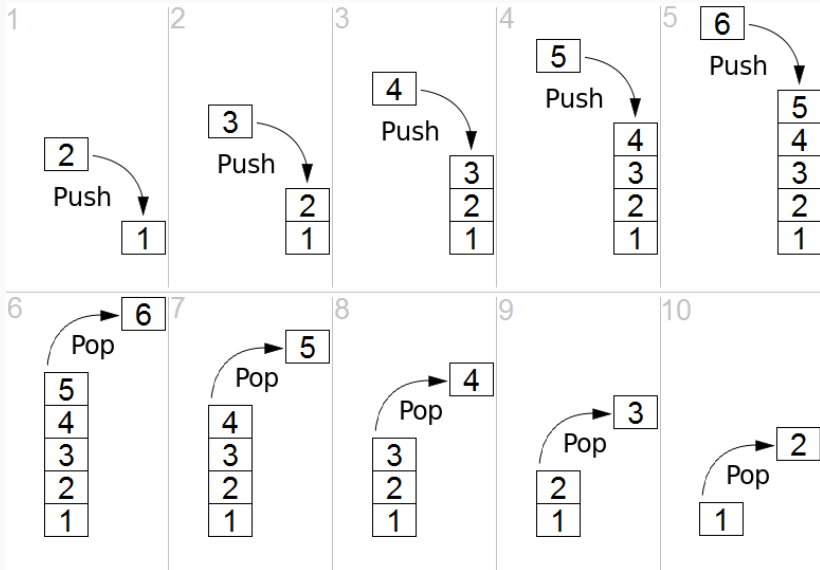
- Существует возможность вызвать функцию внутри самой функции
- Такой вызов функции называется рекурсивным

Пример рекурсии: вычисление факториала

```
int factorial(int n)
{
    if (n == 0)
        return 1;
    else
        return n * factorial(n-1);
}

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

Стек



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

void print_sum(int a, int b)
{
    int p = sum(a, b);
    printf("Sum = %d\n", p);
}

int main()
{
    → int x, y;
      scanf("%d%d", &x, &y);
      print_sum(x, y);
      return 0;
}
```

Стек:

main()

Переменные x и y


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

```
{
```

```
    int s = a + b;
```

```
    return s;
```

```
}
```

```
void print_sum(int a, int b)
```

```
{
```

```
    int p = sum(a, b);
```

```
    printf("Sum = %d\n", p);
```

```
}
```

```
int main()
```

```
{
```

```
    int x, y;
```

```
    scanf("%d%d", &x, &y);
```

```
    print_sum(x, y);
```

```
    return 0;
```

```
}
```



Стек:

main()

Переменные x и y

print_sum(int, int)

Переменная p,
Входные параметры
(если не хватило
регистров)

```

int sum(int a, int b)
{
    int s = a + b;
    return s;
}
void print_sum(int a, int b)
{
    int p = sum(a, b);
    printf("Sum = %d\n", p);
}
int main()
{
    int x, y;
    scanf("%d%d", &x, &y);
    print_sum(x, y);
    return 0;
}

```

→

Стек:

main()

Переменные x и y


print_sum(int, int)

Переменная p,
Входные параметры
(если не хватило
регистров)

sum(int, int)

Переменная s,
Входные параметры
(если не хватило
регистров)

```
int sum(int a, int b)
{
    int s = a + b;
    return s;
}
void print_sum(int a, int b)
{
    int p = sum(a, b);
    printf("Sum = %d\n", p);
}
int main()
{
    int x, y;
    scanf("%d%d", &x, &y);
    print_sum(x, y);
    return 0;
}
```



Стек:

main()

Переменные x и y


print_sum(int, int)

Переменная p,
Входные параметры
(если не хватило
регистров)

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

void print_sum(int a, int b)
{
    int p = sum(a, b);
    printf("Sum = %d\n", p);
}

int main()
{
    int x, y;
    scanf("%d%d", &x, &y);
    print_sum(x, y);
    return 0;
}
```



Стек:

main()

Переменные x и y