Функции

Функция

- Група от операции, които заедно решават определена задача
- Базова модулна единица в езика за програмиране C
- Декларация на функция – прототип на функция
  - return_type function_name(списък с параметри разделени със запетая);

```
int factoriel(int n);
```

- Дефиниция на функция
  - return_type function_name(списък с параметри разделени със запетая) { тяло на функция }
  - Тялото на функцията са операциите, които се изпълняват при извикването на функцията

```
int factoriel(int n)
{
        int res = 1;
        for(int i = 2; i <= n; i++)
                res *= i;
        return res;
}
```

Връщан тип

- Всеки от стандартните типове данни или потребителски тип данни
  - return връщана_стойност; //тази стойност се връща като резултат
  - Връщана стойност:
    - Константа – return 5;
    - Израз – return a*b-c;
    - Извикване на функция – return pow(a,2);
      - Извикваната функция трябва да има същия тип на връщания резултат
    - Комбинация – return (pow(a,2) – 4*b) / 2.0;
- void – функция от този тип не връща резултат
  - Не е задължително присъствието на return
  - return; може да се използва за излизане (за връщане обратно там, откъдето е извикана функцията) от функцията преди края на тялото ѝ

Идентификация

- Три характеристики допринасят за уникалността на функцията
  - Името на функцията
  - Брой параметри
  - Тип на параметрите
- Не може да има две функции, при които да съвпадат и трите характеристики
- void print_line(int val);
- void print_val(int val); //OK, името е различно
- void print_val(float val); //OK, типът на параметъра е различен
- void print_val(float val, int p); //OK, броят на параметрите е различен
- int print_val(int val); //не е OK, връщаният тип не допринася за уникалността

Разполагане на функцията в кода

- Функцията трябва да е декларирана преди функцията, в която се извиква (преди използването ѝ)
- В прототипите на функциите може да не се указват имена на параметрите (но при дефиницията на функцията е задължително)

```
int factoriel(int);
.
.
.

int factoriel(int n)
{
…
}
```

- Най-лесно
  - В началото на файла се декларират прототипите на функциите
  - Функциите се подреждат по подходящ начин (независимо от местата, където се използват)

```
int factoriel (int n);
int main(){
….
f = factoriel(5);
….
}
```

```
int factoriel (int n){ ….. }
int main(){
….
f = factoriel(5);
….
}
```

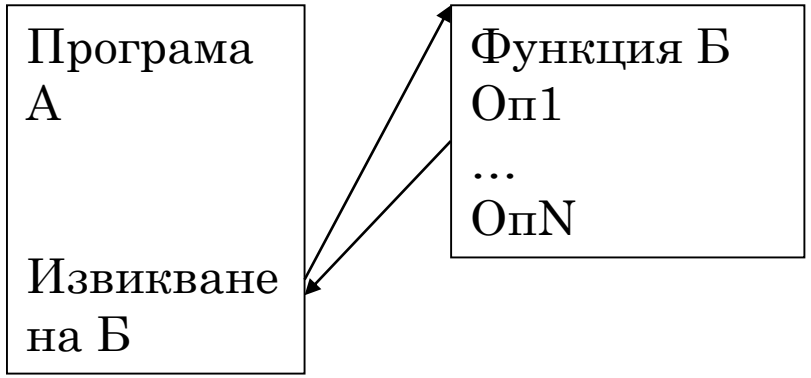
```
int main(){
….
f = factoriel(5);
….
}

int factoriel (int n){ ….. }
```

Грешка при компилация

Извикване на функция

- При извикване на функция, изпълнението на програмата се прехвърля в извиканата функция
- Изпълняват се операциите от тялото на функцията
- След като функцията приключи работа, изпълнението на програмата се връща там, откъдето е била извикана
- Ако функцията връща стойност, на мястото, където е извикана, се получава върнатия от функцията резултат



Извикване на функция

- Функция се извиква с нейното име и в кръгли скоби се задават стойности за параметрите ѝ
  - print_val(5);
  - print_val(var);
- Когато има повече параметри, те се разделят със запетаи
  - print_val(5.3, 2);
- Когато функцията връща резултат, функцията може да участва в изрази
  - fact = factoriel(n);
  - member = factoriel(n) / (float)n;
  - return factoriel(n) / (float)n;
  - print_val(factoriel(n));

Изчисляване на факториел

```
#include <stdio.h>

int factoriel(int);

int main()
{
		int chislo, fact;

		printf("Vavedete chislo:");
		scanf("%d", &chislo);

		fact = factoriel(chislo);
		printf("Faktorielat e %d", fact);
}

int factoriel(int n)
{
		int res = 1, i;

		if (n == 0)
				return 1;

		for (i = 2; i <= n; i++)
				res *= i;

		return res;
}
```

Глобална и локална видимост

Global & Local scope

Глобални променливи

- Декрларират се извън функциите

```
int global_var;
int main()
{
        //операции с global_var
}
void func()
{
        // операции с global_var
}
```

- Стоят в паметта през цялото изпълнение на програмата – заемат място
- Могат да се използват навсякъде в кода (във всички функции)
  - Включително в различни .c / .cpp файлове
    - Ключова дума extern

```
extern int global_var; //extern - тази променлива вече е декларирана на друго място
```

Използване на няколко файла с код

```
//main.cpp

void print_global();

int global_var;
int main()
{
	printf("Vavedete stoinost ");
	scanf("%d", &global_var);
}

//source.cpp

extern int global_var;

void print_global()
{
	printf("globalna stoinost = %d", global_var);
}
```

Локални променливи

- Декларират се между { и }
  - { и } ограничават видимостта на променливите, декларирани между тях

```
void func()
{
		int a;
		a = 5;
		{
			int b;
			b = 4;
		}
		b = 5; //Грешка - b тук не съществува
		a = 6;
}

int main()
{
		printf("%d", a); //Грешка - a е локална променлива за func – тук не съществува
}
```

Глобални vs локални променливи

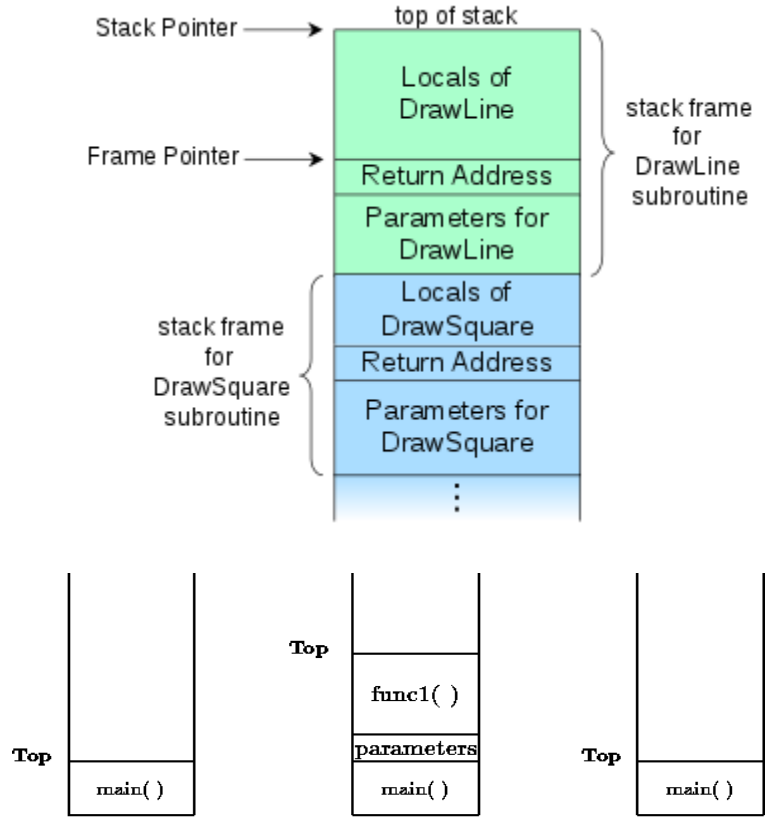
- Внимание: Позволено е декларирането на глобални и локални променливи с еднакви имена – локалната променлива скрива съществуването на глобалната

```
void func(int);

int n;
int main()
{
        scanf("%d", &n);
        func(5);
}
void func(int n)
{
//винаги извежда 5 – локалният параметър n скрива глобалната променлива
        printf("%d", n);
}
```

Стек на извикванията / Call stack

- По време на работата на програмата съдържа информация за текущо активните функции
- При извикване на функция, нейните параметри и локални променливи се записват в стеков запис (stack frame) в стека на извикванията
- При приключване на функцията, нейната стекова рамка се изтрива от стека



Статични променливи (локални)

- Декларират се с ключова дума static
  - static int val;
- Ако при декларацията има инициализация, тя се изпълнява еднократно при първото извикване на функцията
- Остават в стека, когато функцията приключи

```
#include <stdio.h>
void call_counter();

int main()
{
		call_counter();
		call_counter();
}

void call_counter()
{
		static int n = 0;
		n++;
		printf("%d izvikvane\n", n);
}
```

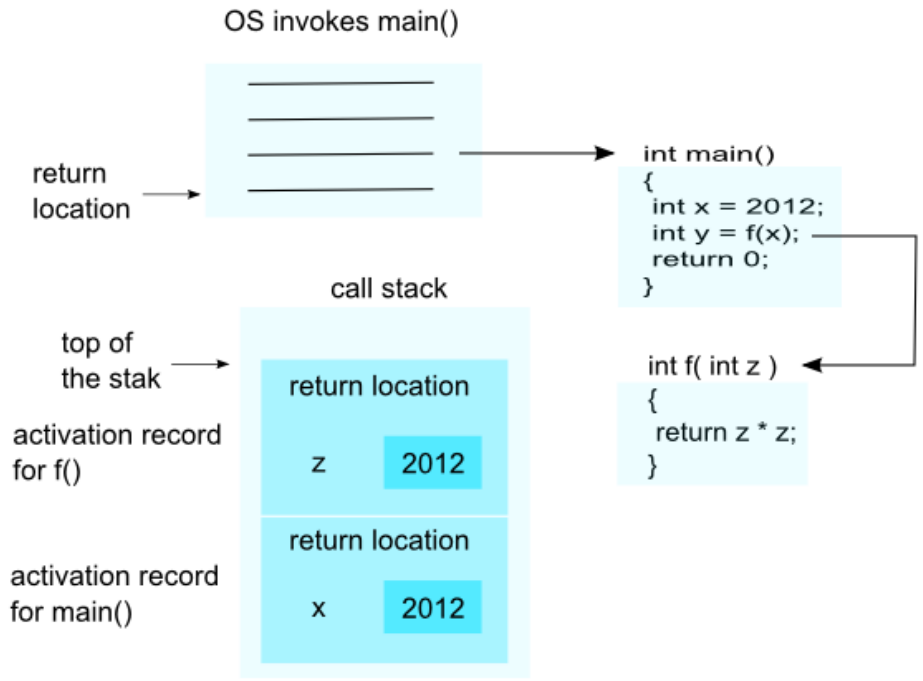
Класове памет

клас_памет тип идентификатор;

Клас памет	**Ключова дума**	**Присъствие в паметта**	**Област**	**Деклариране**
регистров	register	временно	локална	вътре във функциите
статически	static	постоянно	локална	
външен	extern	постоянно	глобална (всички файлове)	вън от функциите
външен статически	static	постоянно	глобална (един файл)	

Предаване на параметър по стойност

- Предава се само стойността на параметъра на функцията
- Ако за стойност на параметъра е подадена променлива, промяна на параметъра във функцията **не** указва влияние върху променливата, от която е зададена стойността на параметъра



Опит (неправилен) за размяна на две стойности с предаване по стойност

```
#include <stdio.h>

void swap(int, int);

int main()
{
		int a = 5, b = 10;
		printf("main: predi swap a = %d, b = %d\n", a, b);
		swap(a,b);
		printf("main: sled swap a = %d, b = %d\n", a, b);
}

void swap(int v1, v2)
{
		int tmp;
		printf("swap: predi razmqna v1 = %d, v2 = %d\n", v1, v2);
		tmp = v1;
		v1 = v2;
		v2 = tmp;
		printf("swap: sled razmqna v1 = %d, v2 = %d\n", v1, v2);
}
```

Стойностите във функцията swap се разменят, но това не се отразява на променливите a и b в main функцията, защото се използва предаване на параметри по стойност. За да могат да се разменят стойностите на a и b трябва да се използва предаване на параметри по адрес (следваща лекция).