Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»

Факультет электротехники и автоматики

Программирование и основы алгоритмизации

Шевченко Алексей Владимирович
Кафедра РАПС

Санкт-Петербург, 2010 г.

```
char
```

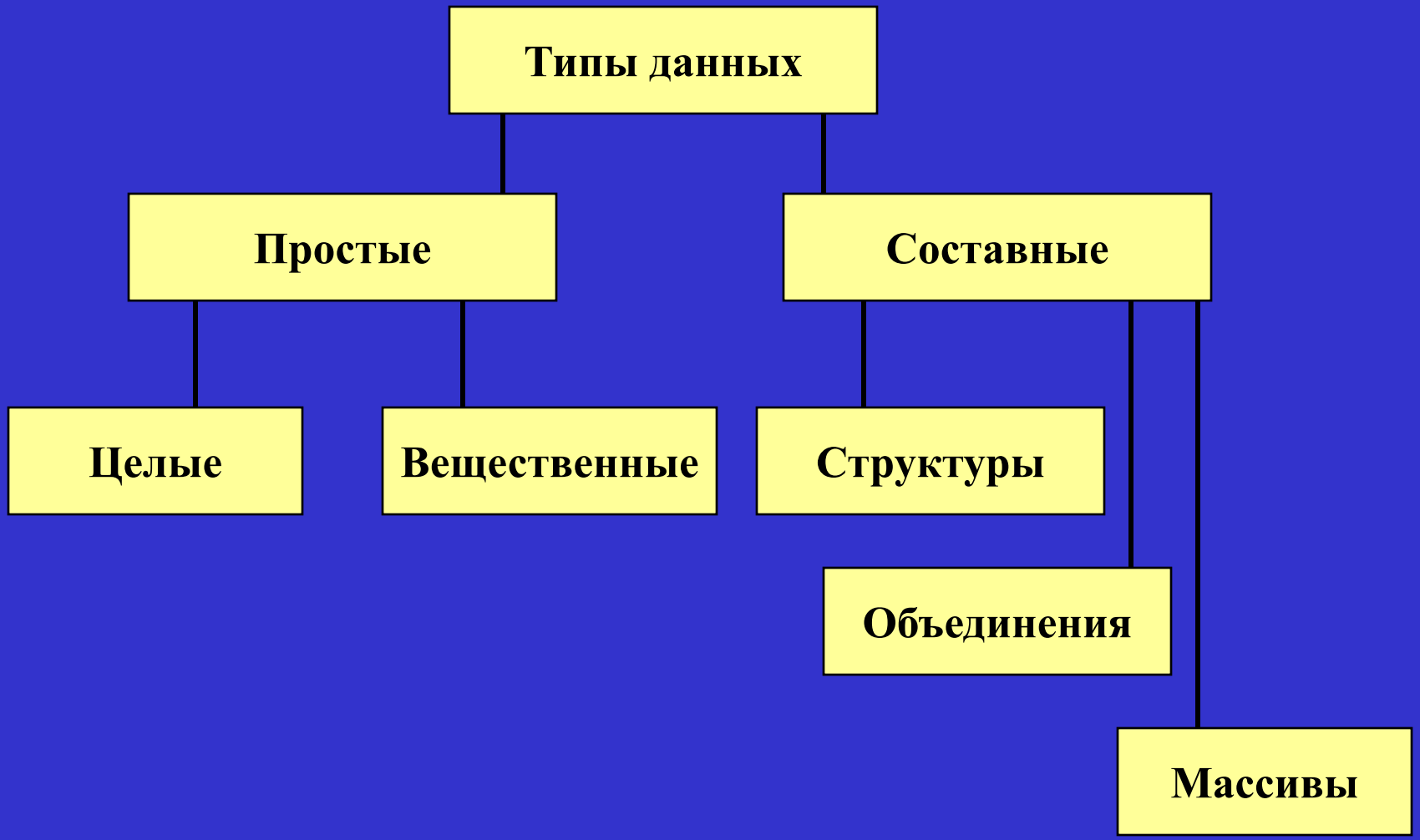
```
double
```

Тема 1. Типы данных, адресация

```
long*
```

```
short
```

Классификация типов данных



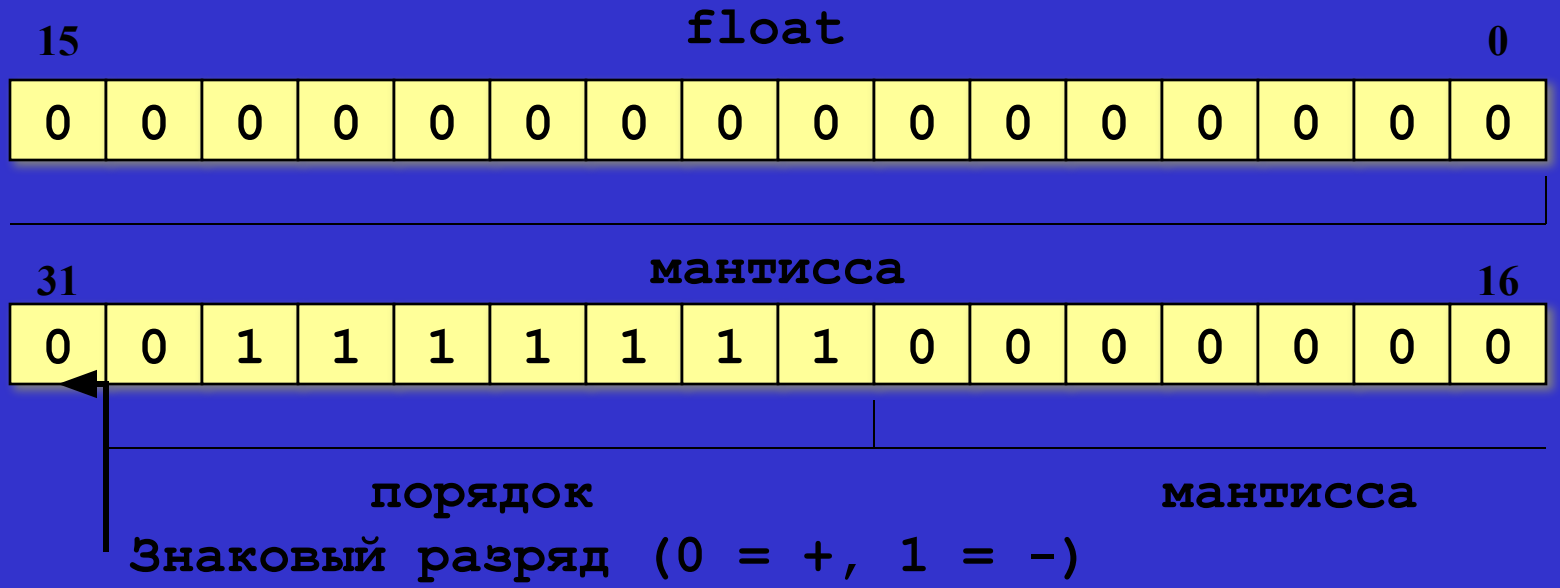
Целые типы данных


```
11111111 = 255
00000000 = 0
```

```
01111111 = 127
00000000 = 0
11111111 = -1
10000000 = -128
```

Тип	Размер	Диапазон
char	1	-128 … 127
unsigned char	1	0 … 255
short	2	-32768 … 32767
unsigned short	2	0 … 65535
long	4	-2^{31} … $2^{31}-1$
unsigned long	4	0 … $2^{32}-1$
int - зависит от платформы		

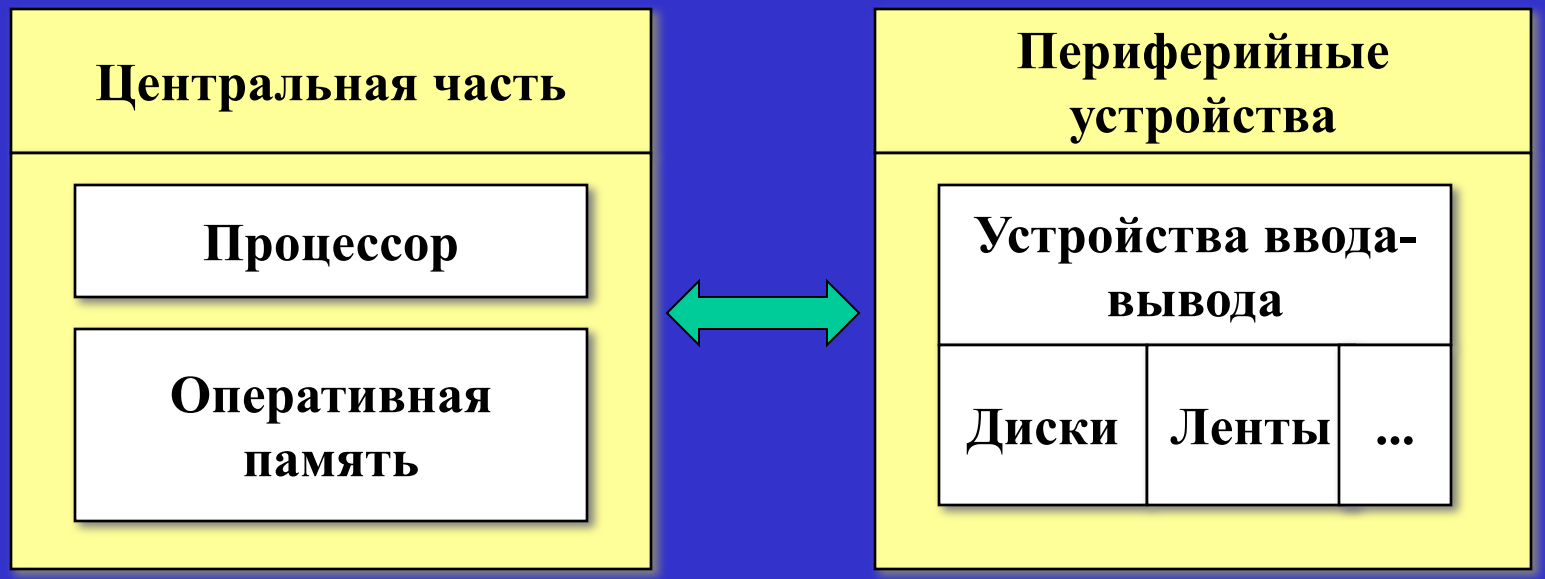
Вещественные типы данных



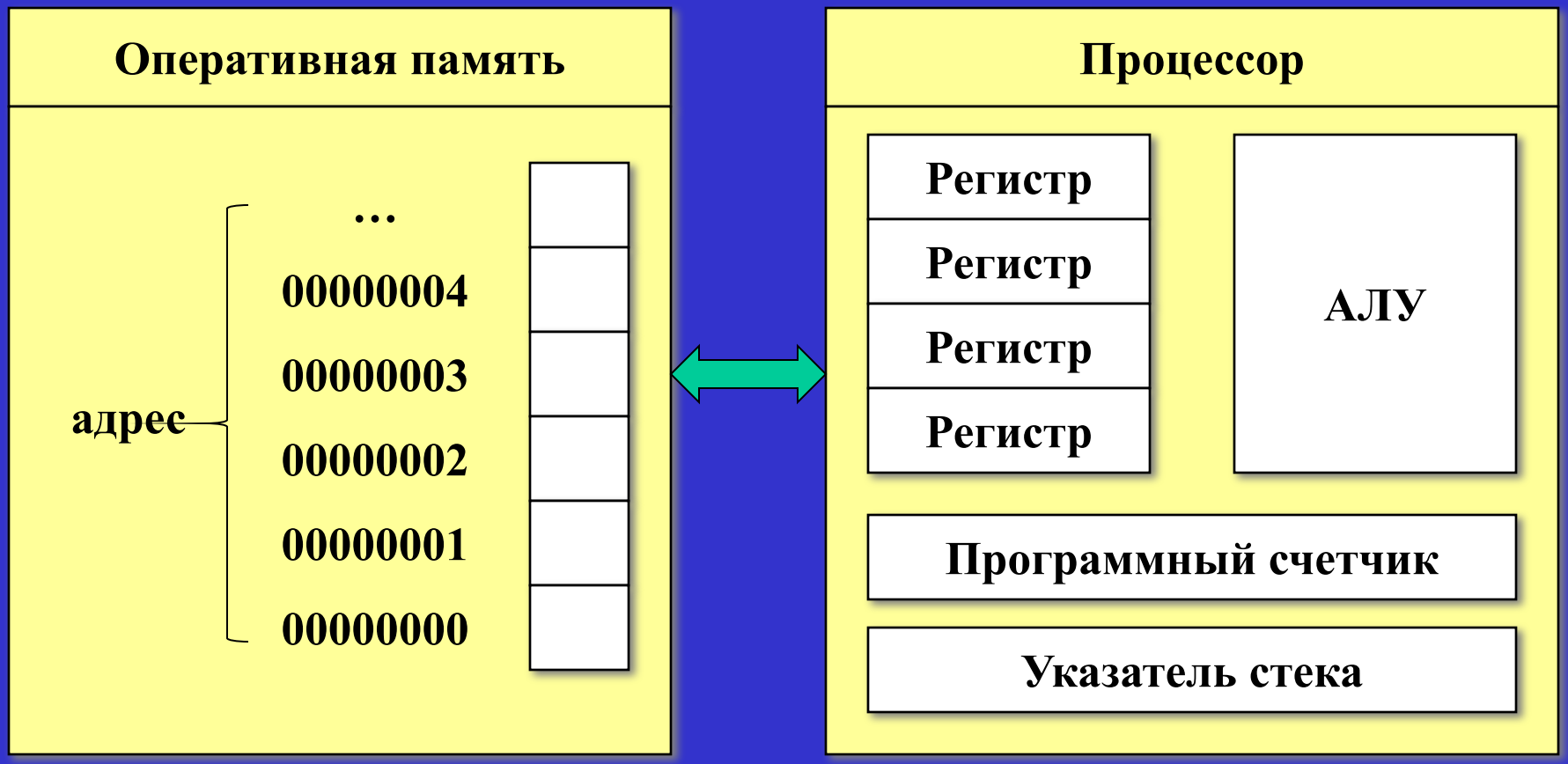
$$\text{Число} = (1+\text{мантисса})*2^{(\text{порядок}-127)}$$

Тип	Размер	Диапазон	Точность
float	4	$3.4*10^{-38}$ … $3.4*10^{38}$	7
double	8	$1.7*10^{-308}$ … $1.7*10^{308}$	15
long double	10	$3.4*10^{-4932}$ … $3.4*10^{4932}$	19

Адресация. Архитектура компьютера

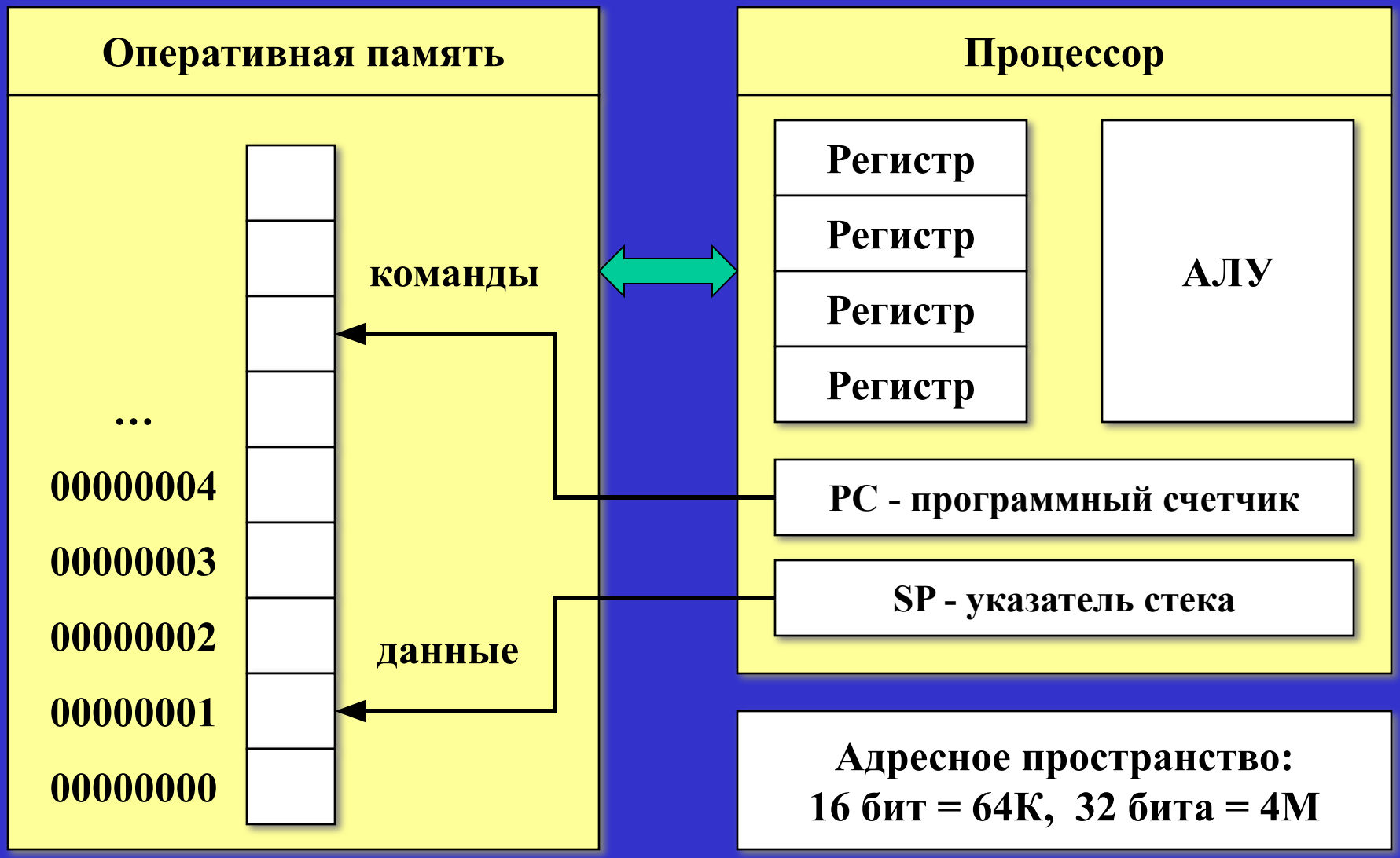


Адресация. Процессор и память

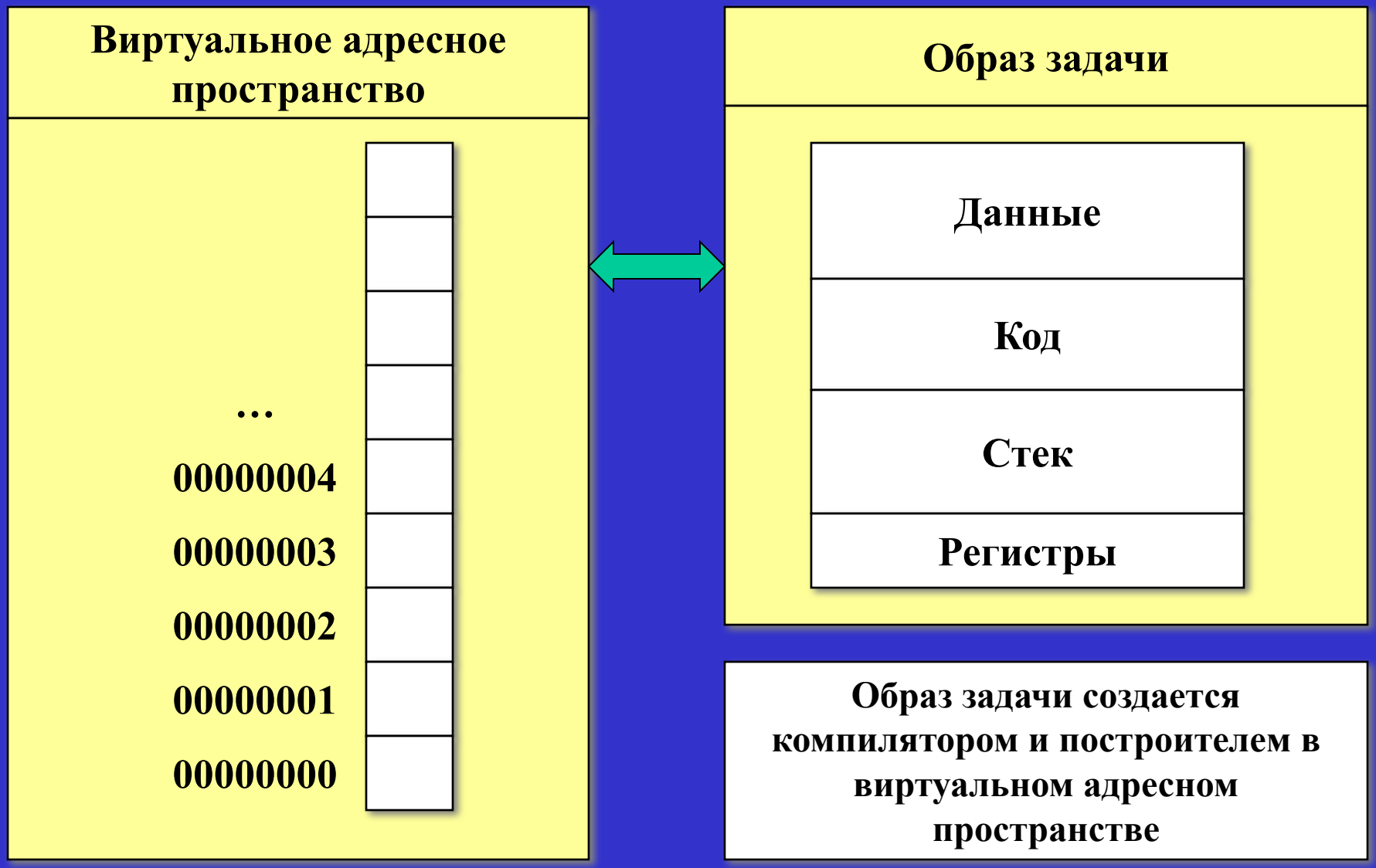


Архитектура процессора: разрядность, система команд

Адресация. Адресное пространство

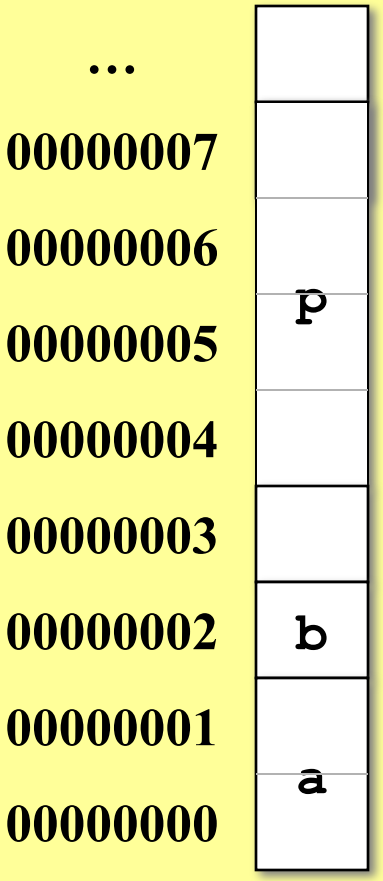


Адресация. Образ задачи



Адресация. Переменные. Указатели

Виртуальное адресное пространство



Текст программы

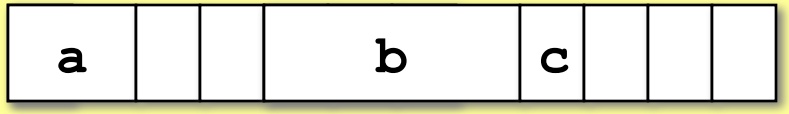
```
short a;
unsigned char b = 3;
short *p = &a;
...
a = 5;
*p = 5;
```

Компилятор размещает переменные и присваивает адреса:

адрес a = 00000000
адрес b = 00000002
адрес p = 00000004
содержимое a = ? / 5
содержимое b = 3
содержимое p = 0

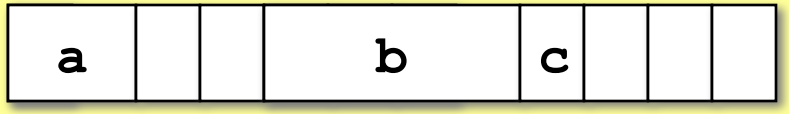
Структуры

A

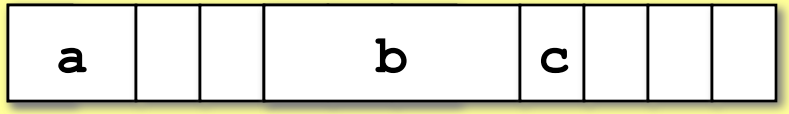


sizeof(A) = 12

a1



a2



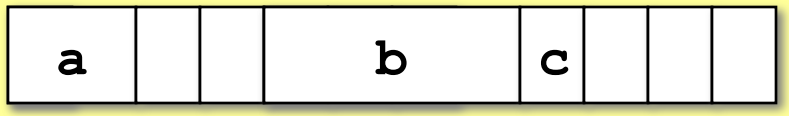
Текст программы

```
typedef struct
{
    short a;
    float b;
    char c;
} A;
…
A a1, a2;
a1.a = -23456;
a1.b = 1234.56;
a1.c = 8;
```

Оператор *typedef* создает новый тип данных. В структурах компилятор применяет выравнивание

Оптимальное размещение данных в структурах

A1



`sizeof(A1) = 12`

A2

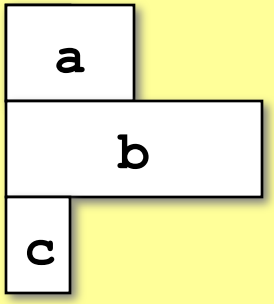
`sizeof(A2) = 8`

Текст программы

```
typedef struct
{
    short a;
    float b;
    char c;
} A1;
…
typedef struct
{
    short a;
    char c;
    float b;
} A2;
```

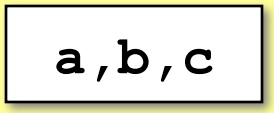
Объединения

B

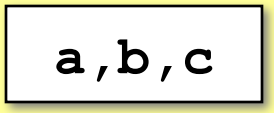


`sizeof(B) = 4`

b1



b2



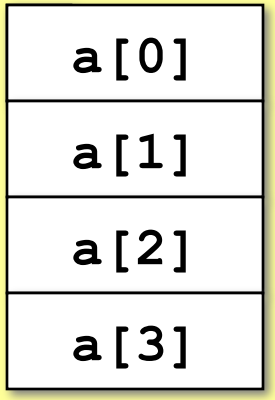
Текст программы

```
typedef union
{
    short a;
    float b;
    char c;
} B;
…
B b1, b2;
b1.a = -23456;
b1.b = 1234.56;
b1.c = 8;
```

В объединениях поля данных перекрываются. Размер равен самому большому полю

Массивы

a



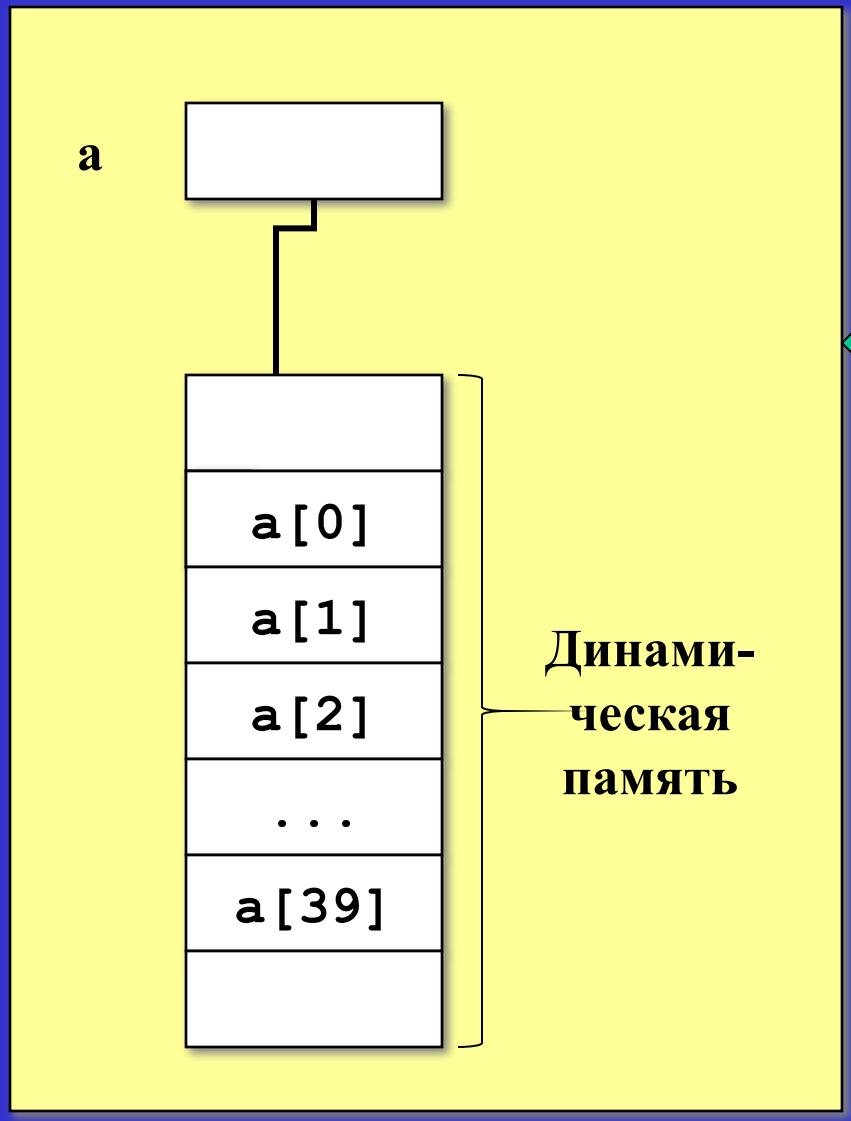

```
sizeof(a) = 16
```

Текст программы

```
int a[4];
…
int *b = a;
…
(a[2] == b[2]) //true
```

Имя массива эквивалентно адресу первого элемента массива

Динамическое выделение памяти под массивы



Текст программы

```
int *a = new int[40];
…
a[12] = 5;
a[39] = 8;
a[40] = … ОПАСНО!!!
…
delete[] a;
```

Имя массива эквивалентно адресу первого элемента массива

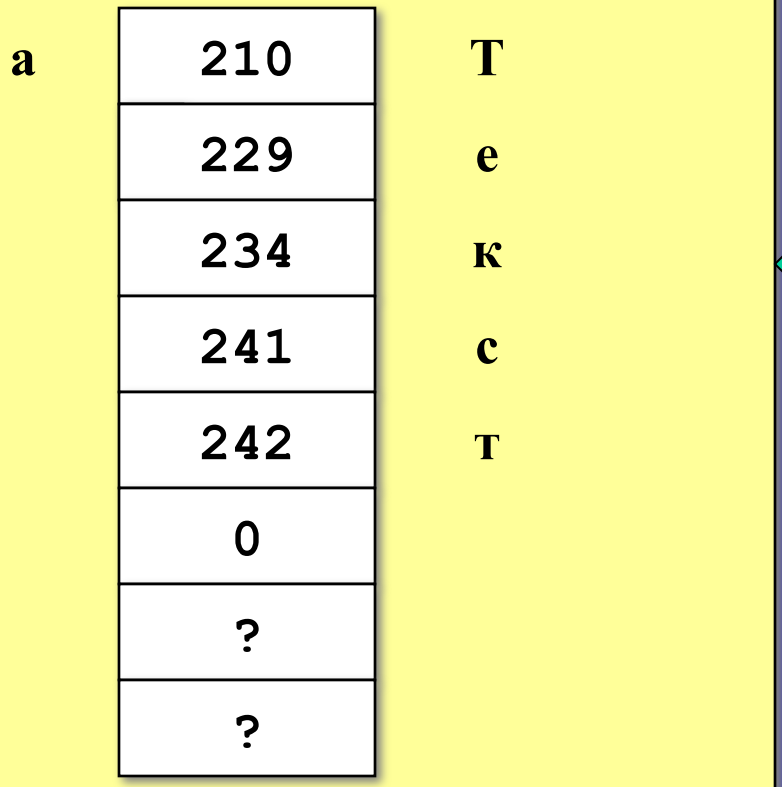
Символы, строки. Кодирование символов

Кодовая таблица Windows (CP-1251)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				©	Ё	§	Є	·		°						
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1																
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2		!	"	#	$	%	&		(	)	*	+				/
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}]	~	□
	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	Ђ	Ѓ	,	ѓ	„	…	†	‡		‰	Љ	‹	Њ	Ќ	Ћ	Џ
	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	ђ	‘	’	"	"		–	—		™	љ	›	њ	ќ	ћ	џ
	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A		Ў	ў	J	¤	Ґ	¦	§	Ё	©	Є	«		-	®	Ї
	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	°	±	І	і	ґ	µ		·	ё	№	є	»	j	S	s	ї
	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	▪ 223
E	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

kod-win1

Символы, строки



Текст программы

```
char a[8];
a[0] = 'Т';
a[1] = 'е';
a[2] = 'к';
a[3] = 'с';
a[4] = 'т';
a[5] = 0;
...
strcpy(a, "Текст");
```

В языках C и C++ строки заканчиваются нулевым байтом

Операции со строками

Текст программы

```
char a[8];          // выделение памяти

char *b = "Текст";  // строковая константа

strcpy(a, b);       // копирование строк

int len = strlen(a); // длина строки

int cmp = strcmp(a, b); // сравнение строк

char* s = strchr(a, 'т');  // поиск символа

char* s = strstr(a, "кс"); // поиск подстроки
```

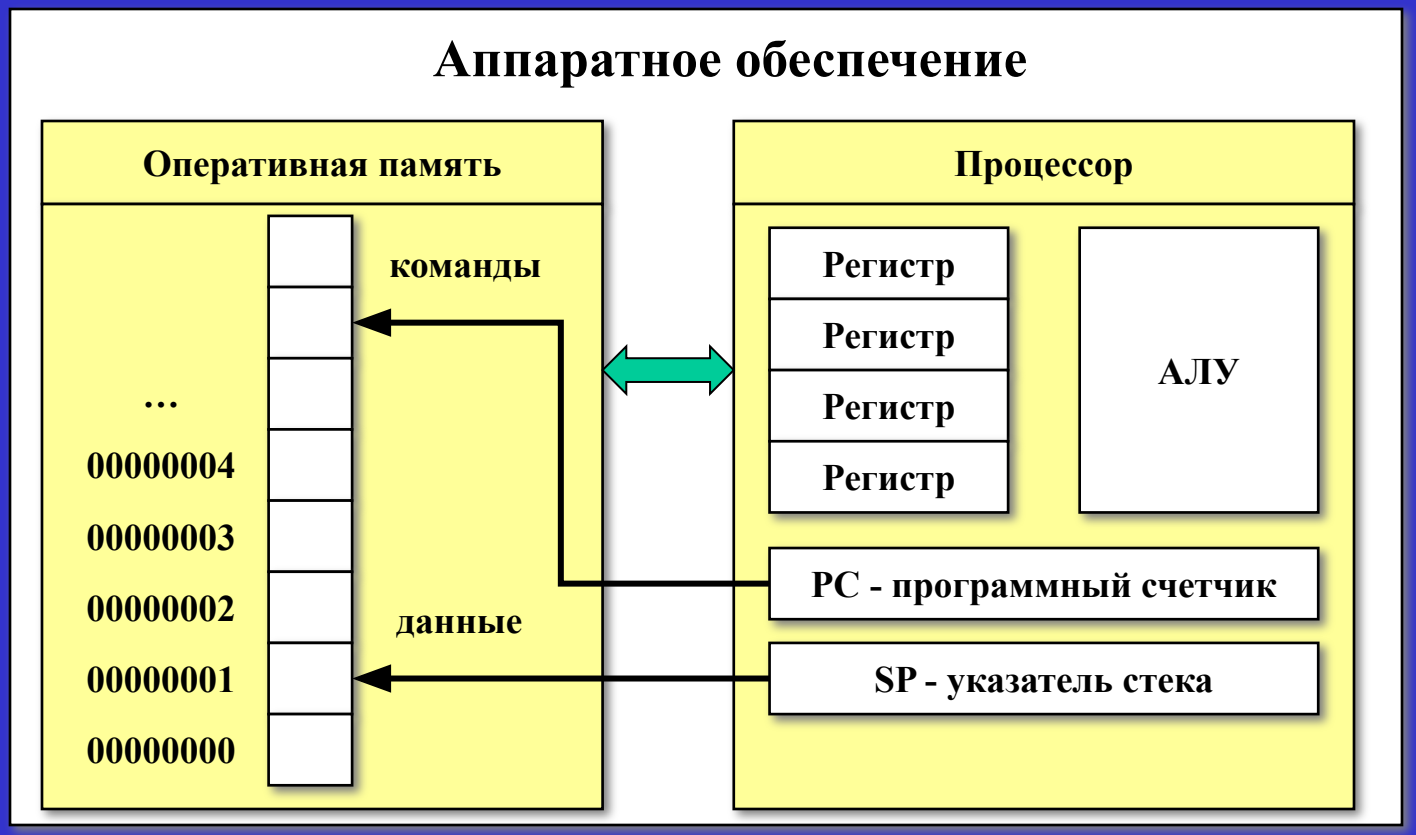
```
double  a;
char  b[64];
```

Тема 2. Переменные, управление памятью

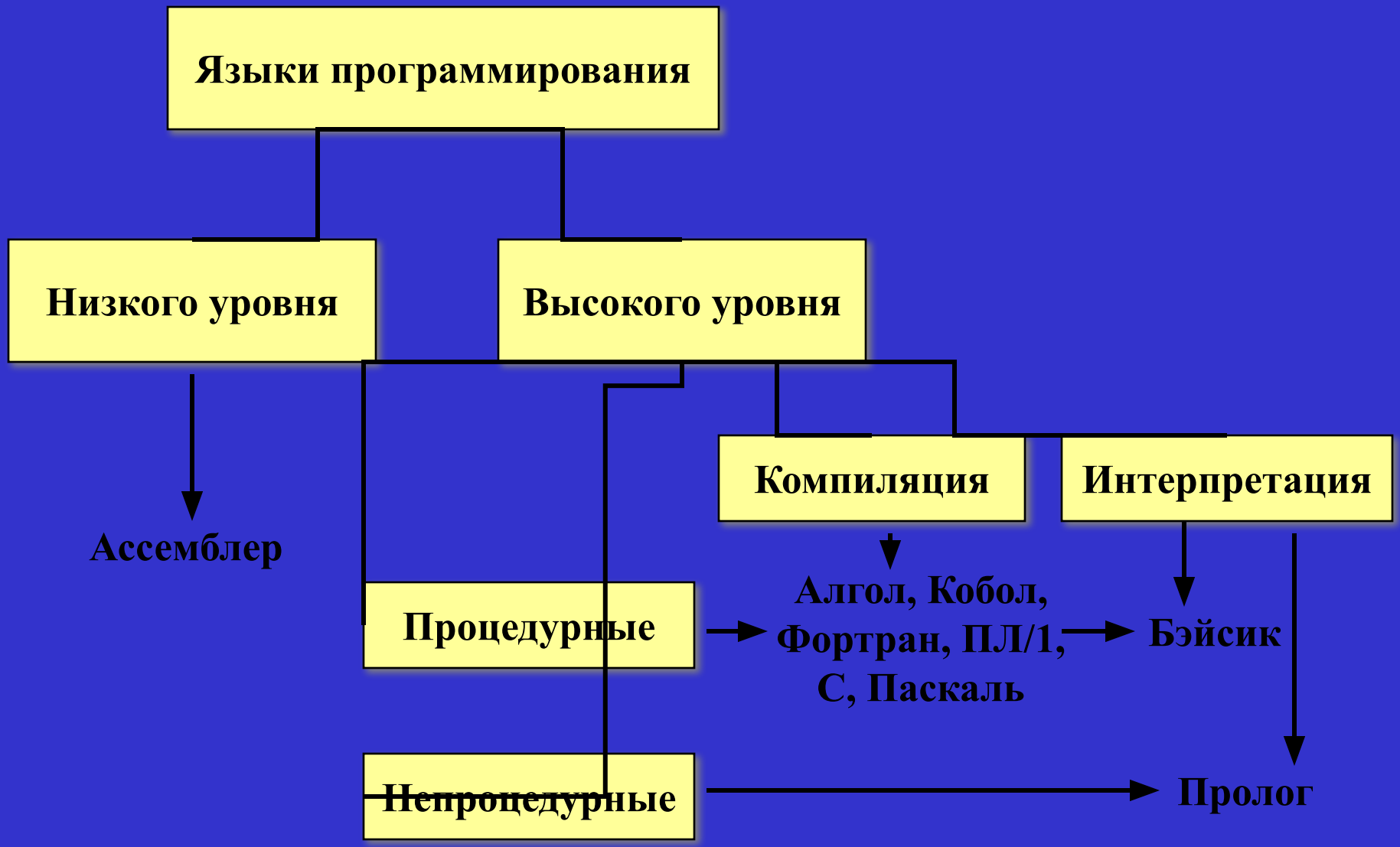
```
long* c = new long(48);
```

Программирование. Задачи

Программное обеспечение



Программирование. Языки

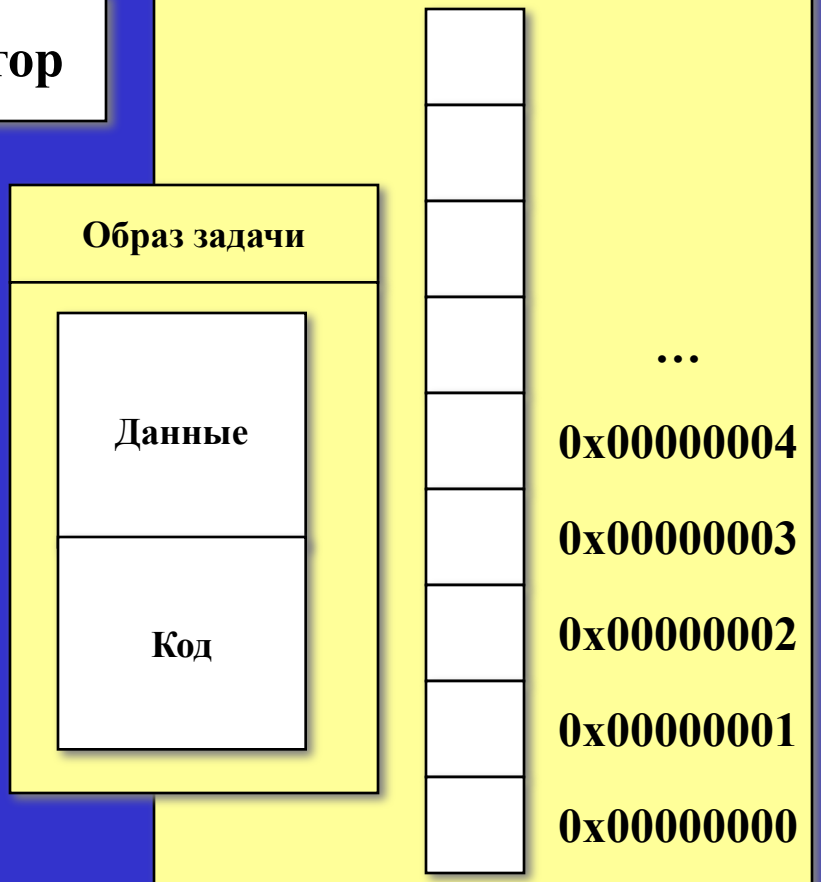


Программирование. Компиляция и построение задачи

Текст программы

```
int a = 5;

int main()
{
    int b = a/5;
    int c = f(b);
}

int f(int a)
{
    return(abs(a));
}
```

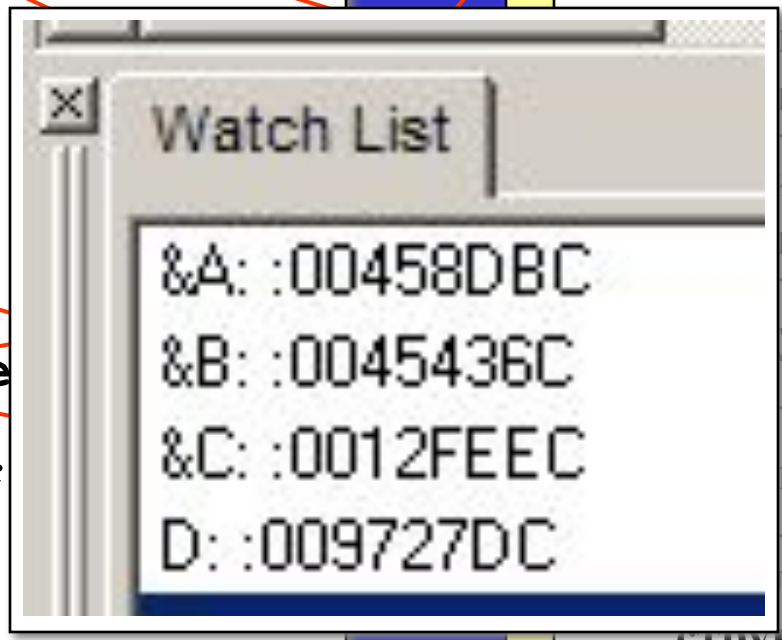
Виртуальное адресное пространство



Размещение переменных в памяти

Текст программы

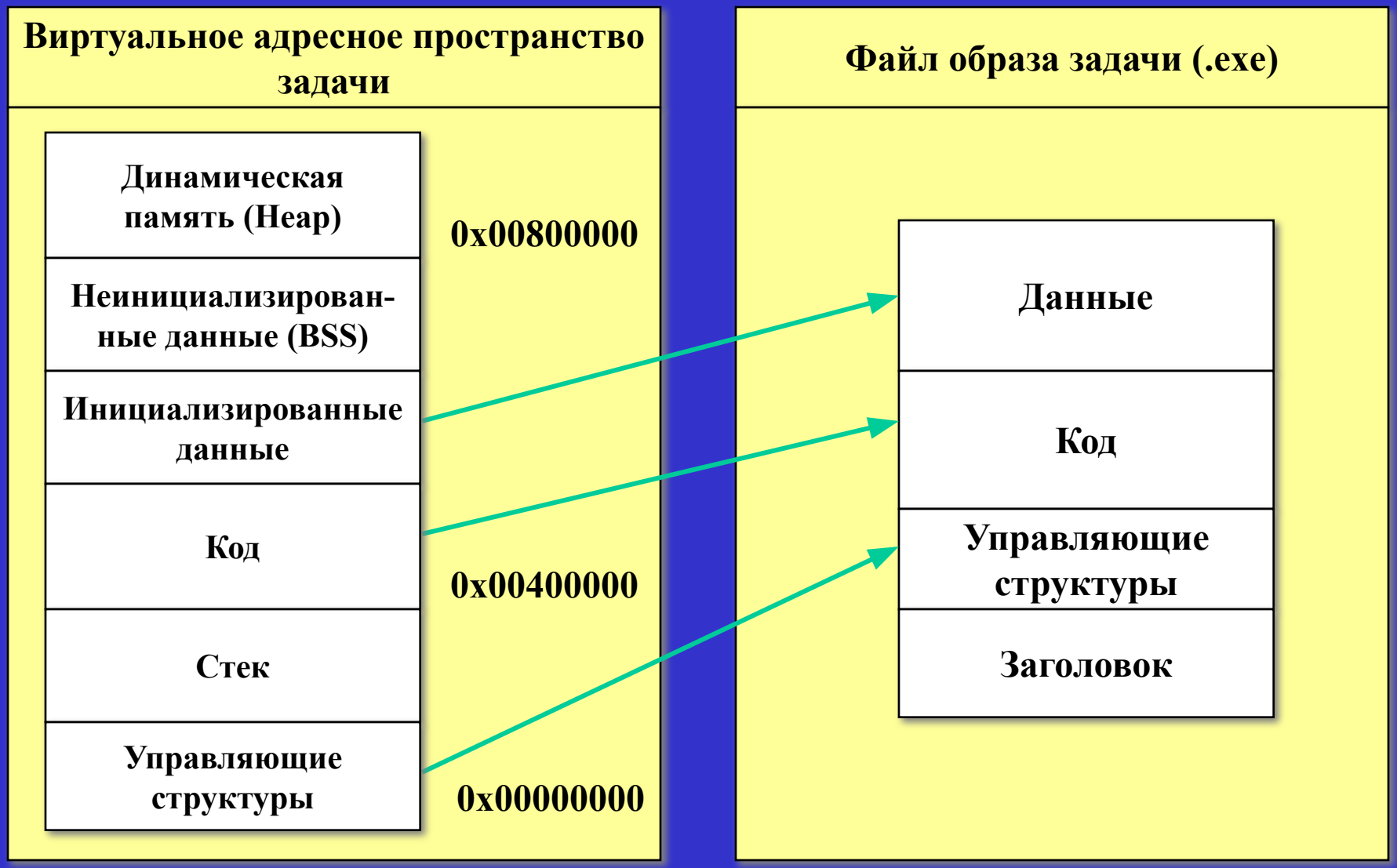
```
int A;

int B = 999;

void f()

{

   int C;

   int *D = ne

   *D = A+B+C;

   ...

}
```



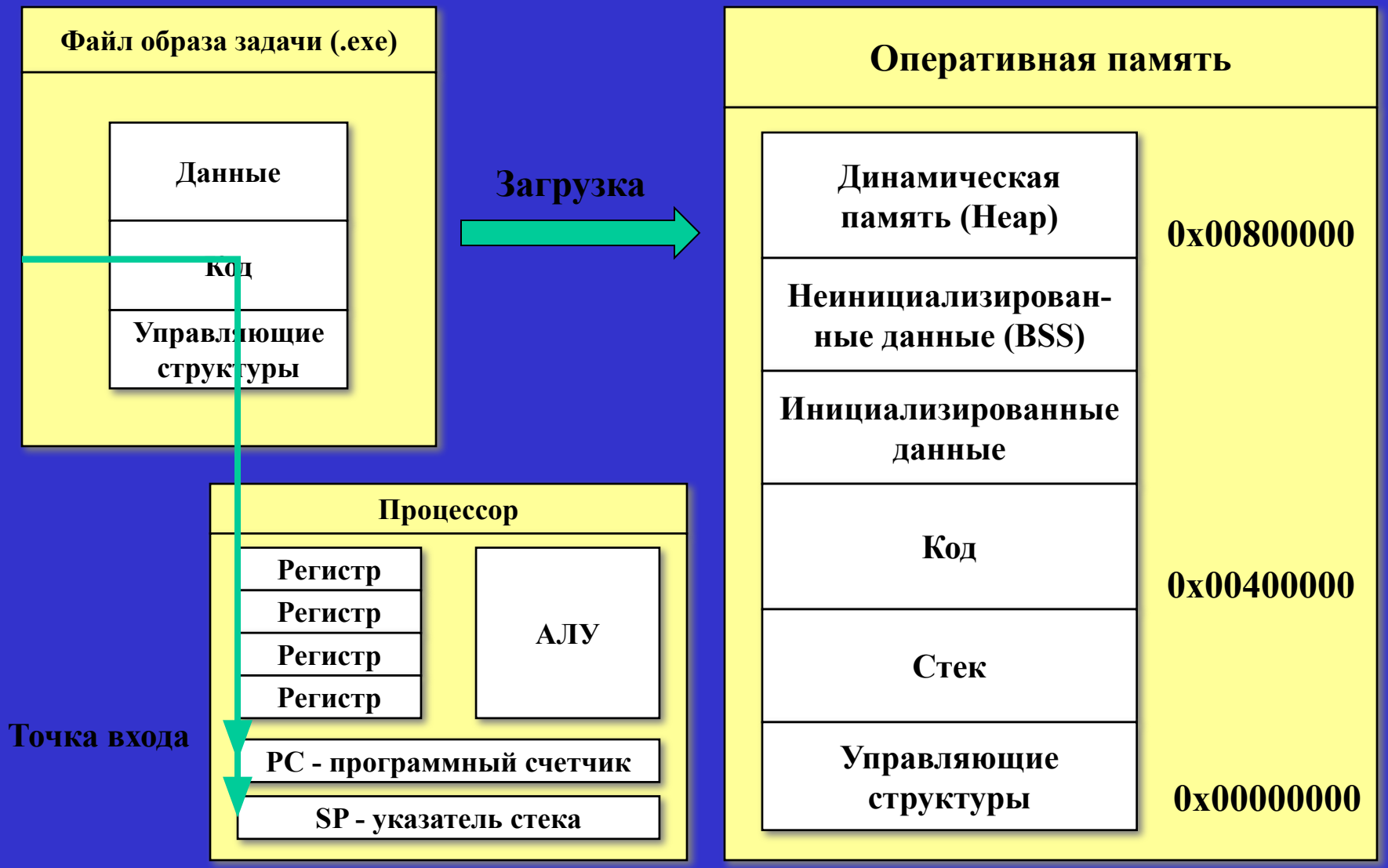
Виртуальное адресное пространство задачи

Динамическая память (Heap)	0x00800000
ициализи- ые данные	
ициализи- ые данные	
код	0x00400000
текст	
няющие структуры	0x00000000

Файл образа задачи



Выполнение задачи



```
#define  MAXCOUNT  100
```

Тема 3. Препроцессор, функции

```
void f(int a, int b);
```

Препроцессор

Текст программы

```
#define MAX 200

int data[MAX];

int main()
{
   for(int i = 0; i < MAX; i++)
      data[i] = ...
}
```

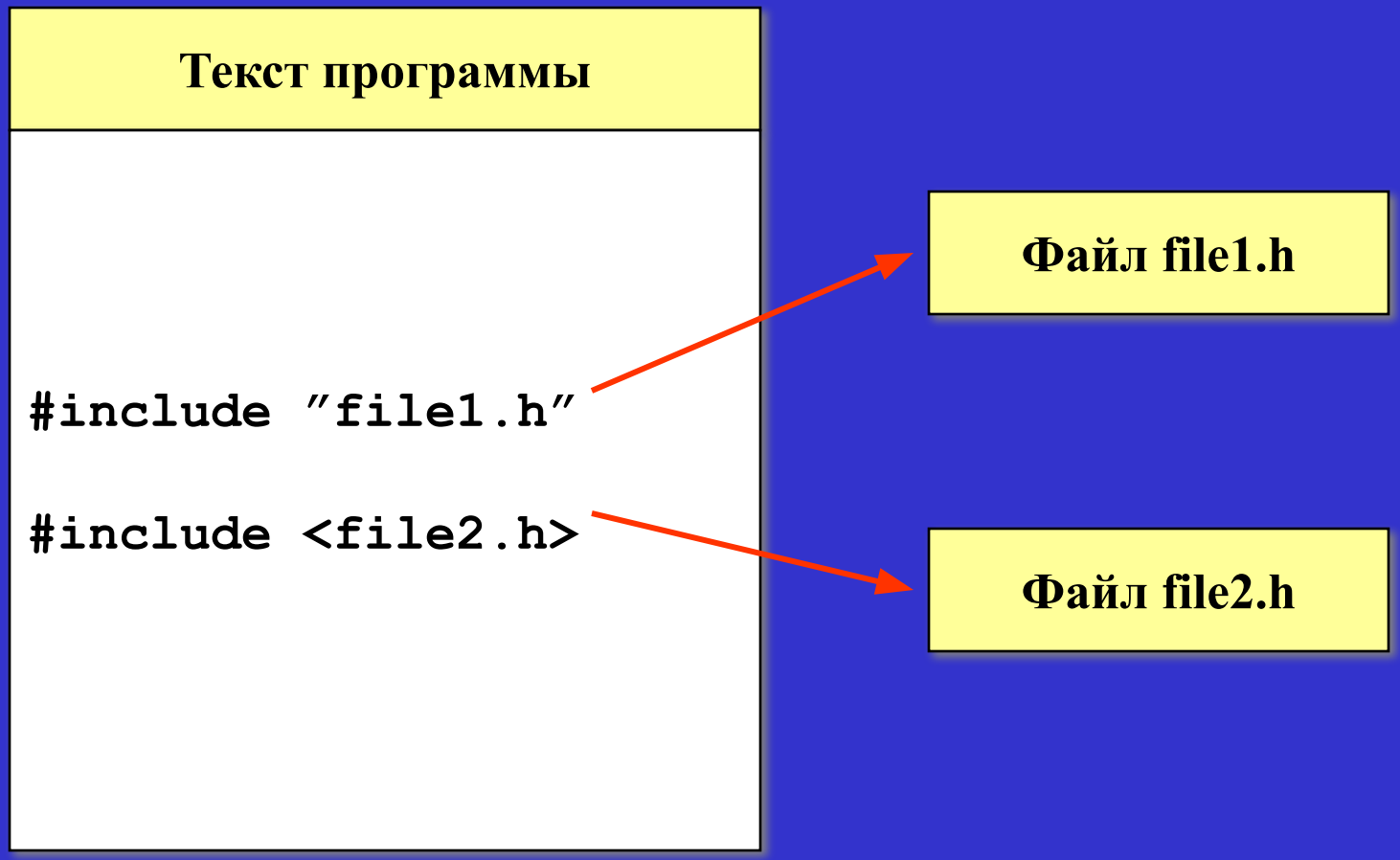
Директивы препроцессора

Директивы препроцессора начинаются с символа #

Текст программы

```
#define MAX 200

#define GETINT(var, edit) \
    var = StrToInt(edit->Text)

#ifdef DEBUG
    ShowMessage("Точка 1");
#endif
```

Включение при компиляции кода из других файлов



Определение символических имен

Текст программы 1

```
#define MAX 200
...
int data[MAX];

for(int i = 0; i < MAX; i++)
   data[i] = 0;
```

Текст программы 2

```
#define Red     0x0000FF
#define Green   0x00FF00
#define Blue    0xFF0000
...
Edit->Color = Red;
```

Стандартные имена компилятора

Текст программы	
__cplusplus	Определено, если компилируется код C++.
__DATE__	Дата начала компиляции текущего файла.
__FILE__	Имя текущего файла.
__FUNC__	Имя текущей функции.
__LINE__	Номер текущей строки.
__STDC__	Определено, если применяется стандарт ANSI.
__TIME__	Время начала компиляции текущего файла.

Макросы

Текст программы

```
#define GETINT(var, edit) \
   var = StrToInt(edit->Text)
...
int rows;
int cols;
...
   GETINT(rows, RowsEdit);
   GETINT(cols, ColsEdit);
```

```
rows = StrToInt(RowsEdit->Text);
cols = StrToInt(ColsEdit->Text);
```

Пример использования макроса

Текст программы 1

```
#define square(a, b) (a*b)
...
int s = square(3+1, 5+1);
```

x = 9

Текст программы 2

```
#define square(a, b) ((a)*(b))
...
int s = square(3+1, 5+1);
```

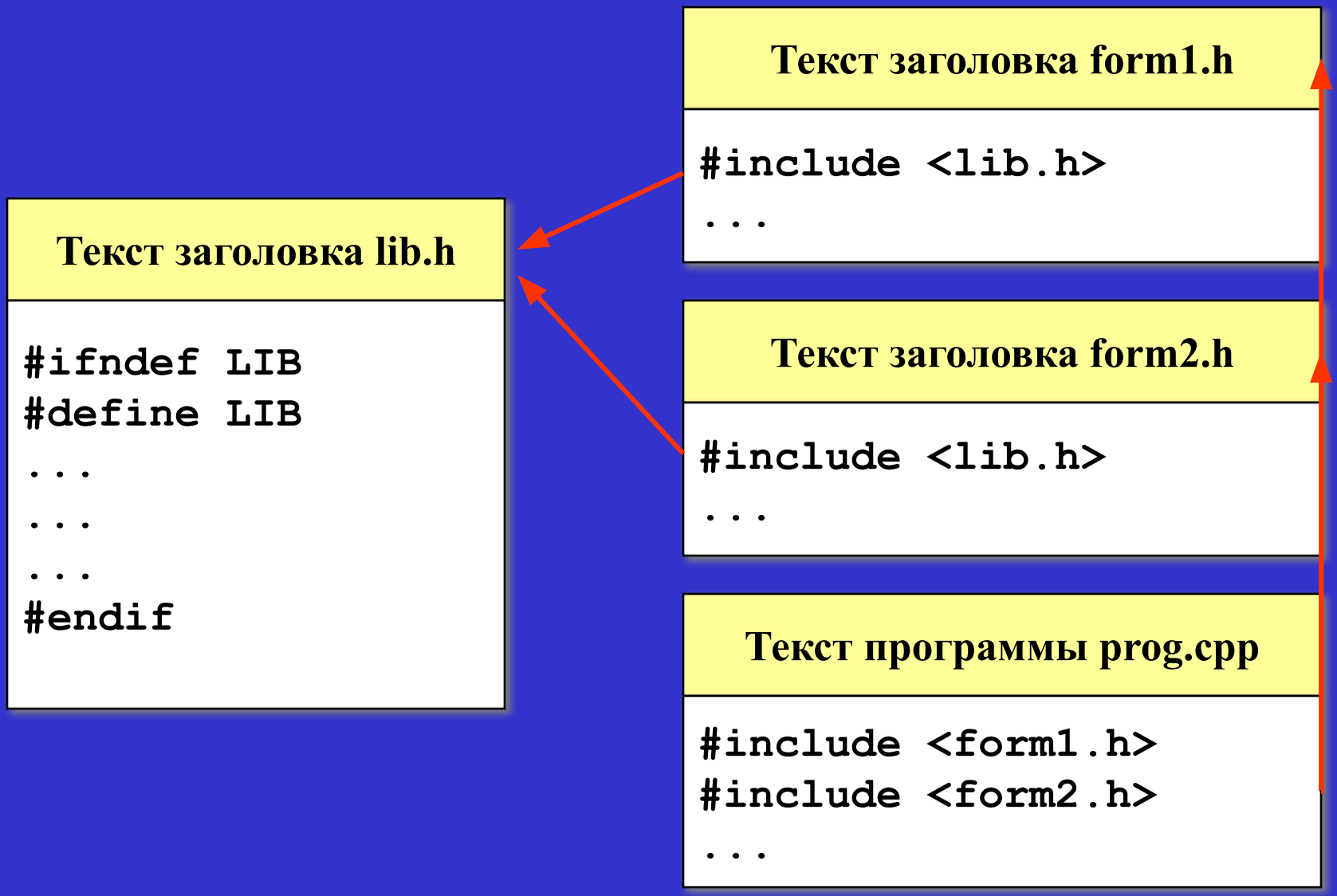
x = 24

Условная трансляция

Текст программы

```
#define DEBUG
#define TRACE

...

long password;

#ifdef DEBUG
#ifdef TRACE
   ShowMessage(”Точка 1”);
#endif
   password = 1;
#else
   GetPassword(password);
#endif
```

Пример использования условной трансляции



Функции

Текст заголовка lib.h

```
int abs(int val);
```

Декларация (прототип функции)

Текст программы prog.cpp

```
#include <lib.h>
...
int a = abs(b);
...
int c = a+abs(d);
```

Текст программы lib.cpp

```
int abs(int val)
{
   if(val < 0)
      val = -val;
   return(val);
}
```

Реализация (тело функции)

Вызов

Декларация функции

Текст заголовка

```
double square(double length, double width);
```

Список параметров

Тип функции | **Имя функции** | **Тип параметра** | **Имя параметра**

```
void func(int a, int b);
```

Функция не возвращает значения

Параметры функции по умолчанию

Текст заголовка

```
int func(int a, int b = 3, int c = 5);
```

Значение параметра по умолчанию

Текст программы

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

int x = func(7, 5, 2);
int x = func(7, 5);
int x = func(7);
int x = func();
```

x = 14

x = 17

x = 15

Ошибка!

Передача параметров в функцию

Текст программы 1

```
void func(int a, int b)
{
   a += b;
}

...

int x = 5;
int y = 3;
func(x, y);
```

По значению

x = 5

Текст программы 2

```
void func(int &a, int &b)
{
   a += b;
}

...

int x = 5;
int y = 3;
func(x, y);
```

По ссылке

x = 8

Функции с переменным числом параметров

Текст программы

```
int sprintf(char *buffer, const char *format, ...);
{
...
   va_list  args;

   va_start(args, format);

   for(int i = 0; i < narg; i++)
   {
      int* par = va_arg(args, int*);
      ...
   }

   va_end(args);
}
```

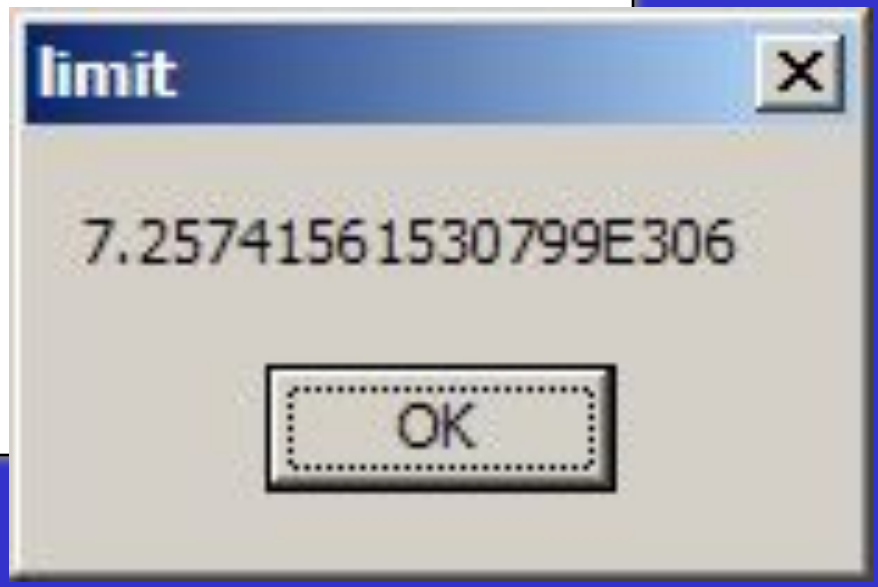
Последний фиксированный параметр

Получение следующего параметра

Рекурсивный вызов функции

Текст программы

```
double factorial(double N)
{
   return(N == 1 ? 1 : N*factorial(N-1));
}

...

ShowMessage(factorial(170));
```



Затраты на вызов функции

Текст программы

```
...

   int a = 5;
   int b = 4;
   int c = square(a, b);

...
```

Параметры, точка возврата, сохраняемые регистры

Стек

```
0x00000005
0x00000004
0x00402768
0x0001238C
```

Сравнение макросов и функций

Макросы	Функции
Быстрота выполнения	Дополнительные затраты времени
Большие затраты памяти	Экономия памяти
Нет контроля типов параметров	Контроль типов параметров

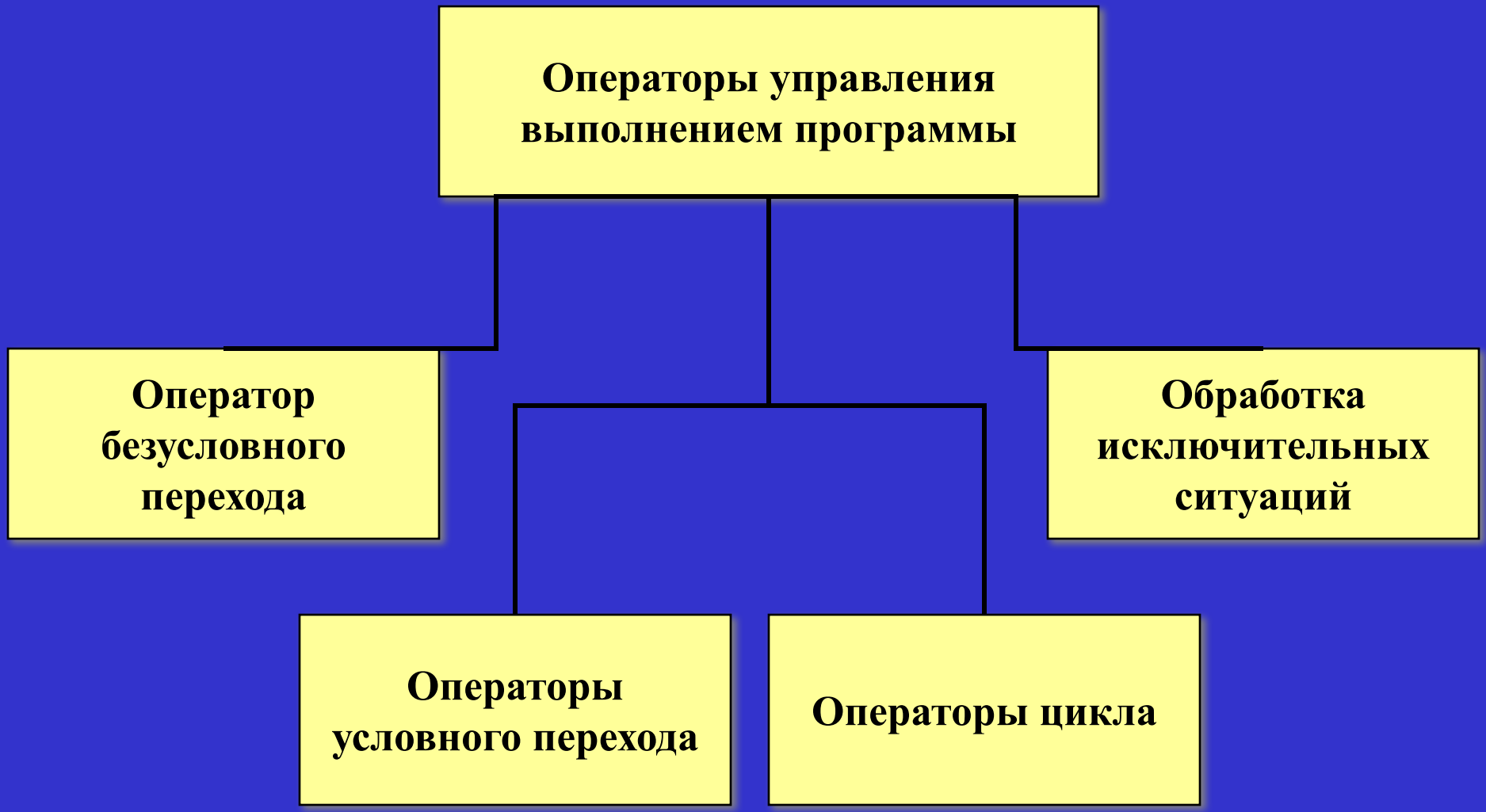
inline - функции, шаблоны

```
for(int i = 0; i < N; i++)
```

Тема 4. Управление выполнением программы

```
try {...} catch(...) {...}
```

Операторы управления выполнением программы



Оператор безусловного перехода

Текст программы

```
...
a = b+c;
goto label;
...
label:
d = e-a;
...
```

Оператор условного перехода if

Текст программы

```
if(a < b)
{
    ...
}
else
{
    ...
}
```

Выражение:
- **логическое (true, false),**
- **арифметическое (не 0 = true, 0 = false)**

Блок «true»

Блок «false»

Примеры оператора if

Текст программы

```
if(a < b)
   a = b;


if(a < b)
   a = b;
else
   a = c;
```

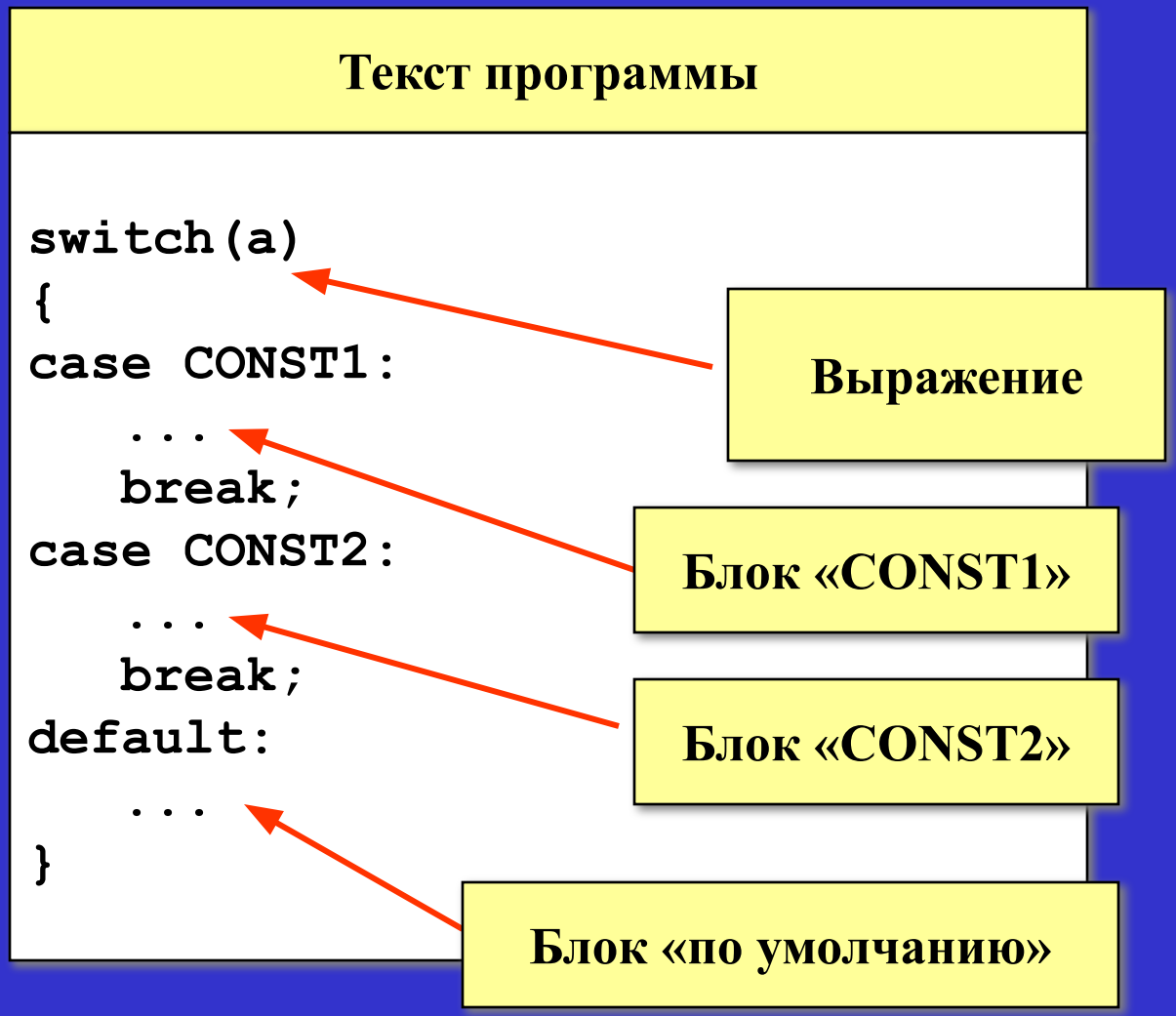
Текст программы

```
if(a < b)
{
   a = b;
   c = b-d;
}
else
{
   a = c;
   d--;
}
```

Текст программы

```
if(a < b)
   a = b;
else
if(a < c)
   a = c;
else
if(a < d)
   a = d;
else
   a = 0;
```

Оператор условного перехода switch



Пример оператора switch

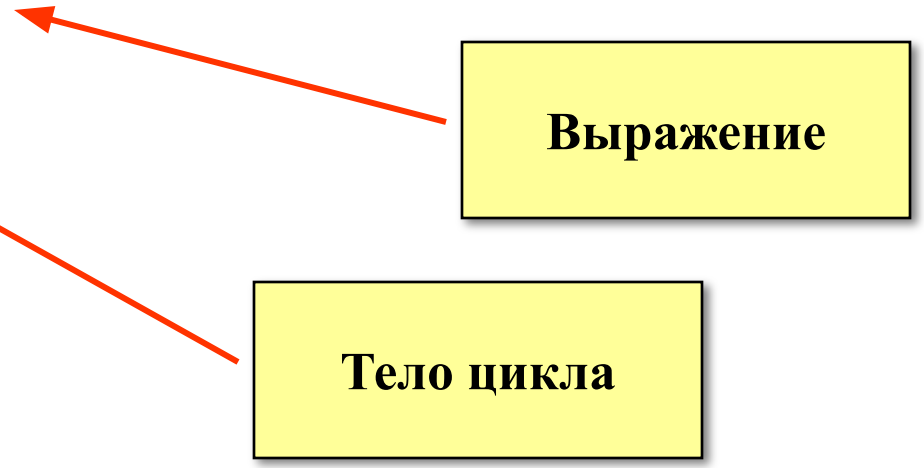
Текст программы

```
char* text;

switch(note)
{
case 5: text = "Отлично";                 break;
case 4: text = "Хорошо";                  break;
case 3: text = "Удовлетворительно";       break;
case 2: text = "Неудовлетворительно"; break;
default:
   text = "Ошибка";
}
```

Оператор цикла while

Текст программы

```
while(a > b)
{
   ...
}
```



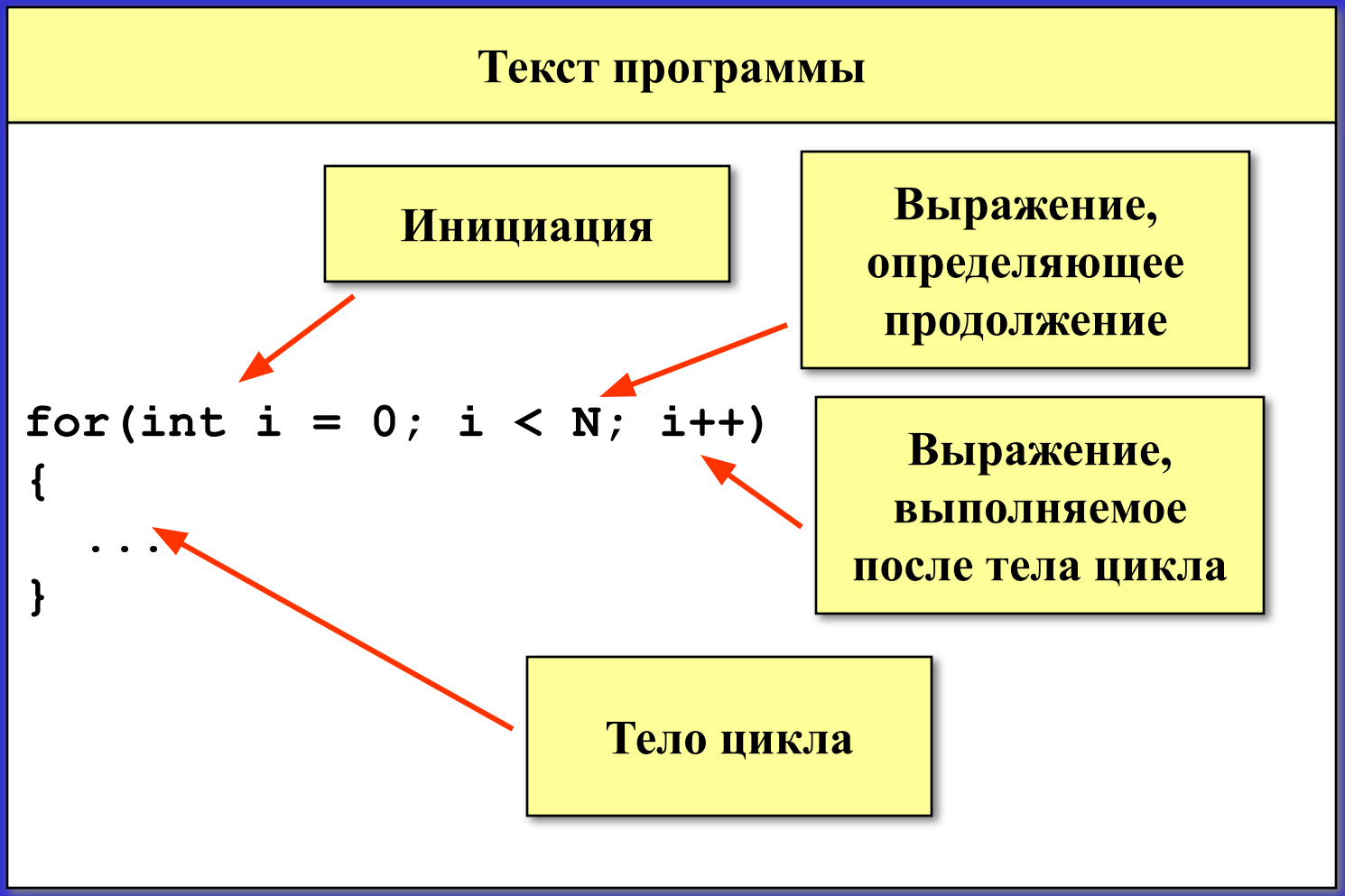
Пример оператора while

Текст программы

```
int a = 10;

while(a--)
{
    ShowMessage("Сколько можно повторять!");
}
```

Оператор цикла for

Текст программы



Пример оператора for

Текст программы

```
int factorial(int N)
{
   int f = 1;

   for(int i = 1; i <= N; i++)
      f *= i;

   return(f);
}
```

Оператор цикла do

Текст программы

```
do
{
   ...
}
while(a < b)
```

Тело цикла

Выражение, определяющее продолжение

Операторы break, continue и return

Текст программы

```
while(true)
{
    ...

    if(...) continue;

    ...

    if(...) break;

    ...

    if(...) return;
}
```

Переход к концу цикла

Выход из цикла

Выход из функции