Домашнее задание № 2

Study-Inf/1 курс/ПИ/ Информатика и программирование

Study-Inf/1 курс/БИ/ Информатика

Домашние задания и самостоятельная работа

Часть 2. Сдается бумажный вариант

2. СИНТАКСИС И АЛФАВИТ ЯЗЫКА СИ

2.1. Алфавит языка

Для образования лексических частей языка (лексем) и связей между ними используются:

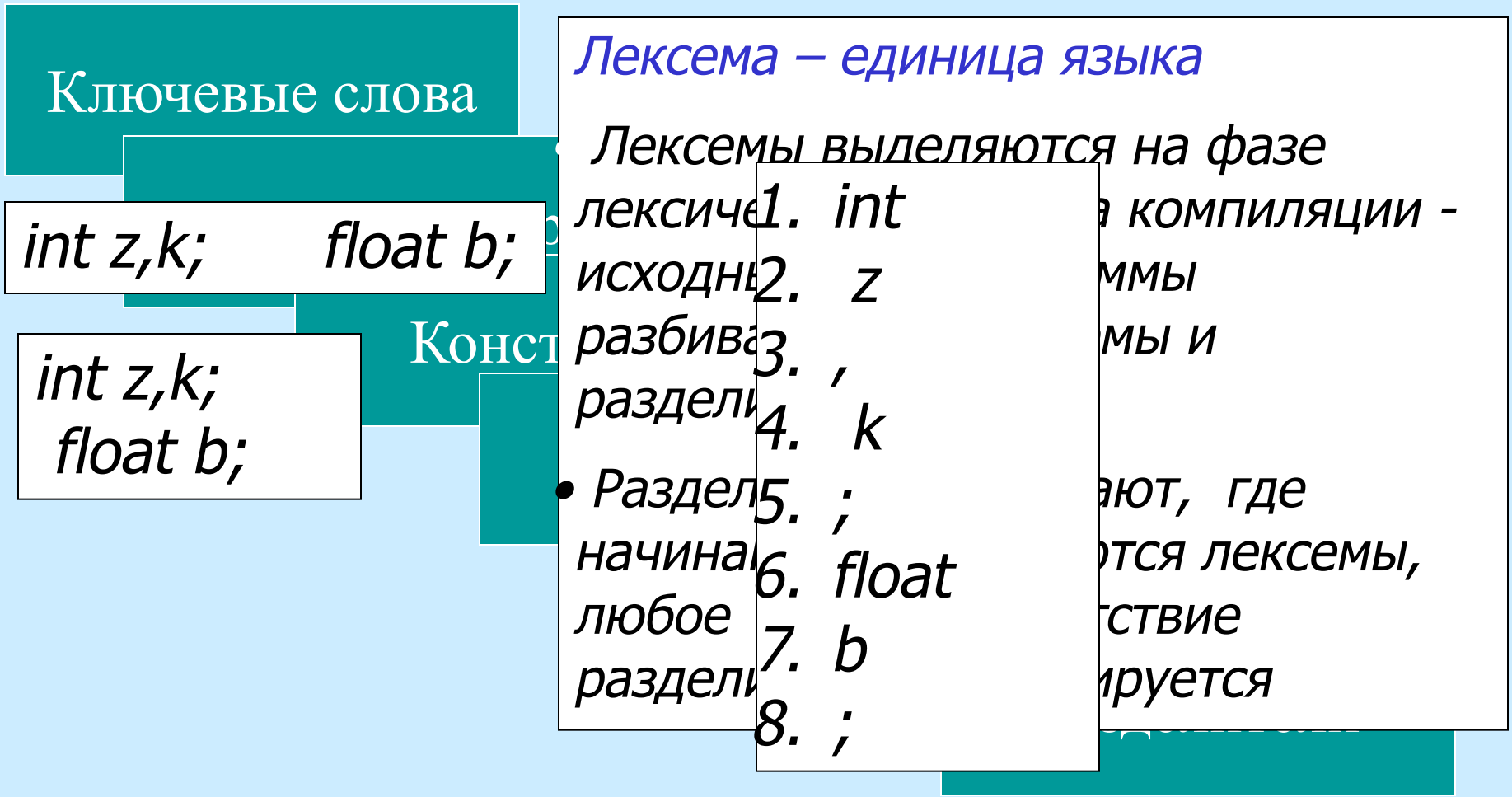
все символы латинского алфавита

цифры

специальные знаки ! @ % $ & * ( ) - + \ / | {} [ ] . , _ ~ " ` # :

2.2. Синтаксис языка

2.2.1. Лексемы языка



2.2.2. Ключевые слова

Ключевые слова - это слова, зарезервированные для специального предназначения и их нельзя использовать как имена идентификаторов.

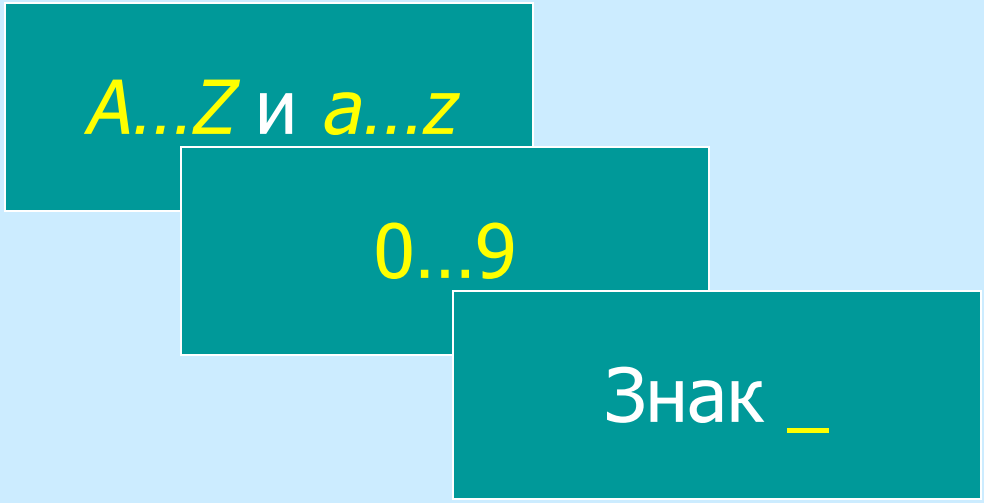
Таблица 1

asm	case	cdecl	const
auto	catch	char	continue
break	_cdecl	class	_cs

default	*for*	*operator*	*_ss*
delete	*friend*	*_pascal*	*static*
do	*goto*	*pascal*	*struct*
double	*huge*	*private*	*switch*
_ds	*if*	*protected*	*template*
else	*inline*	*public*	*this*
enum	*int*	*register*	*typedef*
_es	*interrupt*	*return*	*union*
_export	*_loadds*	*_saveregs*	*unsigned*
extern	*long*	*_seg*	*virtual*
_far	*_near*	*short*	*void*
far	*near*	*signed*	*volatile*
float	*new*	*sizeof*	*while*

2.2.3. Идентификаторы

Идентификаторы - это произвольные имена любой длины для классов, объектов, функций, переменных, типов данных, определенных пользователем и т.д.



Ограничения

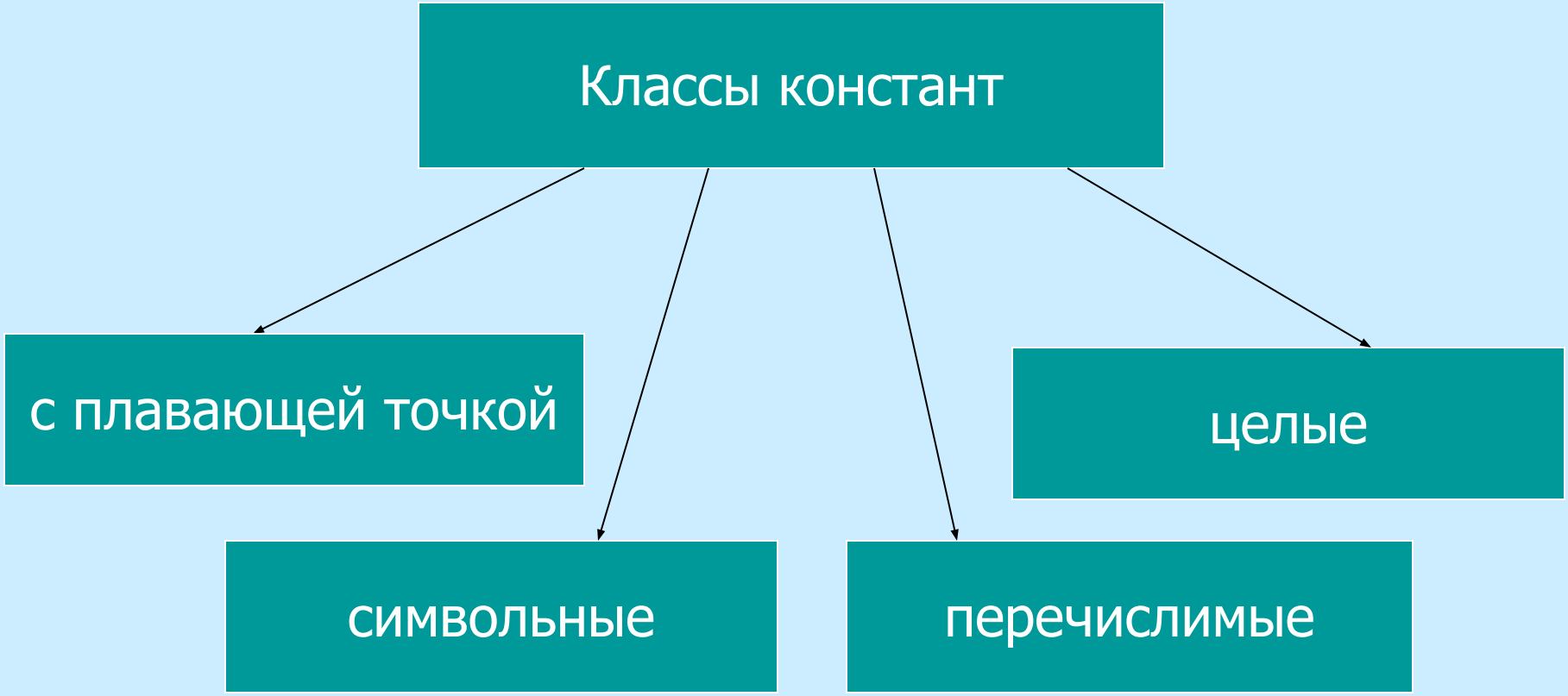
Первый символ должен быть буквой или _.

Распознаются только первые 32 символа

Си – регистро-зависимый язык

Sum, sum и suM - разные идентификаторы

2.2.4. Константы



Целые константы

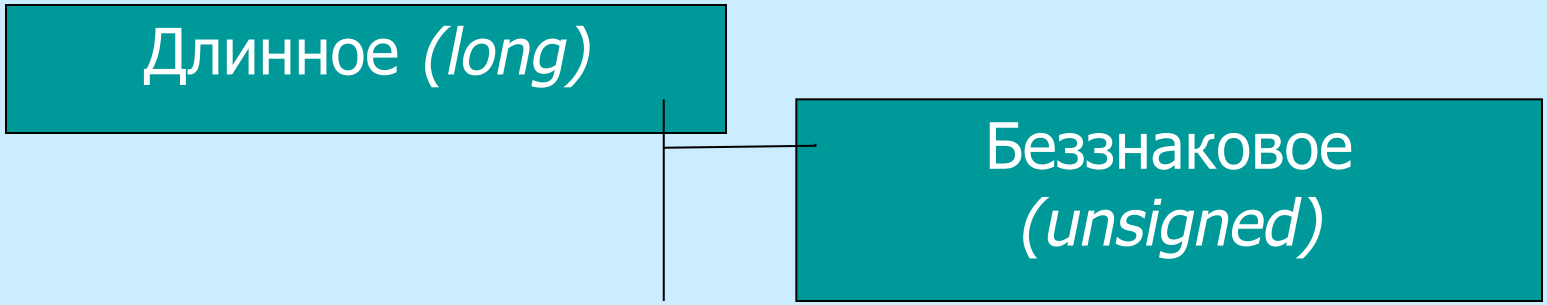
- *Десятичные*
- *Восьмеричные*
- *Шестнадцатеричные*

от 0 до 4294967295

- записывается лидирующий 0
- при содержании в записи числа 9 и 8 генерируется ошибка
- Максимальное значение - 037777777777

- начинаются с 0x (0X)
- максимальное значение 0xFFFFFFFF

Любая целая константа может заканчиваться суффиксами *L* и (или) *U*



Константы в инициализации переменных:

```
int z = 10  /* десятичное 10*/
int x = 010 /*  десятичное 8 */
int p = 0XF /*  десятичное 15 */
```

Константы в выражениях:

```
long m = 2*100L /*  умножение на десятичную константу типа long int */
unsigned n = 2*100U /*  умножение на десятичную константу типа unsigned int */
```

Символьные константы

Символьная константа - это один или более символов, заключенные в апострофы.

используется для обозначения ESC последовательности:

- *отображение некоторых графических символов.*
- *представление ASCII символов или управляющего кода*

```
'A'
'='
'\n'
```

Тип
char

Тип
int

Таблица 2. *ESC* последовательности в Си

Последовательность	Числовое обозначение	Символ	Что выполняет
\a	0x07	BEL	сигнал
\b	0x08	BS	удаление предыдущего символа
\f	0x0C	FF	перевод страницы
\n	0x0A	LF	новая строка (пустая строка)
\r	0x0D	CR	возврат каретки
\t	0x09	HT	табуляция (горизонтальная)
\v	0x0B	VT	табуляция (вертикальная)
\\	0x5c	\	обратный \
\'	0x27	'	одиночная кавычка (апостроф)

Таблица 2. *Продолжение*

Последовательность	Числовое обозначение	Символ	Что выполняет
\"	0x22	"	двойная кавычка
\?	0x3F	?	знак вопроса
\O	любой	O	строка до трех восьмеричных цифр
\xH	любой	H	строка шестнадцатеричных цифр
\XH	любой	H	

Константы с плавающей точкой

Примеры вещественных констант:

3.25e4 - $3.25 \cdot 10^{4}$

.05 - 0.05

1. - 1.0

-1.25 - -1.25

3e-8 - $3.0 \cdot 10^{-8}$

2.5E+6- $2.5 \cdot 10^{6}$

2.2.5. Литеральные строки

Литеральная строка - это массив символов, записанный как последовательность любого числа символов внутри кавычек: "это пример литеральной строки".

строка "\n\"Hello !\"\n"

Вывод на экран:

"Hello !"

- используются для обработки фиксированных последовательностей символов
- символы внутри кавычек могут включать ESC последовательности
- хранится в памяти как заданная последовательность символов и завершается нулевым символом ('\0')
- нулевая (пустая)строка хранится как символ '\0'.
- пустая строка обозначается ""

2.2.6. Операторы

Оператор - это лексема, которая выполняет некоторые вычисления, когда применяется к переменной или к другому объекту в выражении.

- Унарные
- Бинарные
- Один тернарный

Таблица 3. Унарные операторы

Код оператора	Название	Результат операции
&	адресный оператор	*&x* - адрес переменной *x*
*	*оператор косвенной адресации*	**x – значение, расположенное по адресу x*
+	унарный плюс	+5 – положительная константа
-	унарный минус	- 4 – отрицательная константа, *-x* – значение переменной *x* с обратным знаком

Таблица 3. Продолжение

Код оператора	Название	Результат операции
~	*побитовое отрицание*	*~x – побитовое отрицание переменной x*
!	*логическое отрицание*	*!x принимает значение false (лжи), если x имеет ненулевое (истинное) значение и наоборот*
++	префиксное/ постфиксное увеличение	*int x = 5; ++x;* увеличит *x* на единицу; *int x = 5; x++;* увеличит *x* на единицу

Таблица 4. Бинарные операторы

Код оператора	Название	Результат операции
+	*бинарный плюс*	*вычисление суммы* *int x = 2,y = 1,z;* *z = x+y;*
-	*бинарный минус*	*вычисление разности* *int x = 2,y = 1,z;* *z = x-y;*

*	умножение	вычисление произведения
		int x = 2,y = 1,z;
		z = x*y;

Таблица 4. Продолжение

Код оператора	Название	Результат операции
/	*Деление*	*вычисление частного,* *int x = 12,y = 2,z;* *z = x/y;*
%	*остаток от деления*	*вычисление остатка от деления* *int x = 12,y = 7,z;* *z = x%y;*
<<	сдвиг влево	вычисление побитового сдвига влево int x = 12,y = 2,z; z = x << y;

Таблица 4. Продолжение

Код оператора	Название	Результат операции
>>	*сдвиг вправо*	*вычисление побитового сдвига вправо* *int x = 12,y = 2, z;* *z = x >> y;*
&	*побитовое AND (И)*	*вычисление конъюнкции* *int x = 12,y = 2, z = x & y;*
^	побитовое XOR (исключающее или)	вычисление сложения по модулю 2 int x = 12,y = 2, z = x ^ y;

Таблица 4. Продолжение

Код оператора	Название	Результат операции
|	*побитовое OR (ИЛИ)*	*вычисление дизъюнкции* *int x = 12,y = 2, z = x | y;*
&&	логическое AND (И)	проверка условий, связанных логическим и
||	логическое OR (ИЛИ)	проверка условий, связанных логическим ИЛИ

Таблица 4. Продолжение

Код оператора	Название	Результат операции
=	*присваивание*	*присвоить переменной заданное значение или значение другой переменной*
*=	*присвоить произведение*	*выражение x*=5 эквивалентно выражению x = x*5*
+=,	%=, <<=, >>=	&=, ^=, |=, -=, /=

Выражение, использующее операторы отношения, в результате работы принимает значение ***true***, если отношение истинно если же отношение ложное, выражение принимает значение ***false***.

Таблица 4. Продолжение

Код оператора	Название	Результат операции
<	*меньше чем*	*x<y, x меньше y*
<=	*меньше или равно чем*	*x<=y, x меньше или равно y*
>	*больше чем*	*x>y, x больше y*
>=	*больше или равно чем*	*x>=y, x больше или равно y*

Таблица 4. Продолжение

Код оператора	Название	Результат операции
==	*равно*	*x==y, x равно y*
!=	*не равно*	*x!=y, x не равно y*
	Операторы выбора компонент	
.	прямой селектор компоненты	x .k - компонента k переменной x (применяется при работе с объектами и структурами)
->	непрямой селектор компоненты	x->k – компонента k указателя x (применяется при работе с указателями на структуры или объекты)

Таблица 4. Продолжение

Код оператора	Название	Результат операции
,	*оператор перечисления*	*выполнить разделенные оператором перечисления слева направо, например: y+=5,x-=4,y+=x;*

Тернарный оператор *A ? X : Y*

Если истинно отношение A, то выполняются действия X; иначе выполняются действия Y.

```
z  = (x<y) ? x+15:y-25;
```

2.2.7. Знаки пунктуации

[ ] ( ) { } , ; : … * = #.

- [ ] - *указывают список индексов одномерного или многомерного массива:*

```
char  word[] = "Пример строки";  /* строка символов.*/
float mat[3][4];  /* матрица вещественных символов, имеющая три строки и четыре столбца. */
int x[3];  /* целочисленный массив из трех элементов. */
```

- () - выделяют групповое выражение, условное выражение, используются для изменения обычного порядка выполнения операторов и указывают на вызов функции и параметры функции

```
d = (a+b)*x;     /* указывают на порядок действий */
if (x==z) x+=z;  /* используются в условных выражениях */
matrix();        /* вызов функции matrix без аргументов */
int change(int x,int y); /* объявление функции с аргументами */
```

- { } - указывают на начало и конец составного оператора:

```
for(int i =0; i<10;i++)
{ x ++; y--; }
```

- *, - разделяет элементы списка аргументов функции, используется для перечисления действий, вместо составного оператора.*

```
void func(int n, float f, char ch);
for(int i =0; i<10;i++)
  x ++, y--;
```

- ; - указывает на конец оператора

  Любое правильное выражение (включая пустое выражение) должно заканчиваться ";".

- ":“ - указывает помеченный оператор.

```
switch (a) { /* пример использования множественного
    выбора */
        case 1: puts("One");
                break;
        case 2: puts("Two");
                break;
      ...
      default:  puts("None of the above!");
                break;
      }
```

- * - указывает на создание указателя на тип

```
char * str; /* указатель на символ */
int ** x ;   /* указатель на указатель на int */
```

- # - указывает на директиву препроцессора, используется для замещения и объединения лексем во время фазы препроцессора.

3. Основные типы данных

3.1. Простые типы

char	1 байт	Символы, целые числа от 0 до 255.
int	4 байта	Целые числа от -2147483648 до 2147483647.
float	4 байта	Числа с плавающей точкой от $\pm(3.4 \cdot 10^{-38}$ до $3.4 \cdot 10^{38})$
double	8 байт	Числа с плавающей точкой от $\pm(1.7 \cdot 10^{-308}$ до $1.7 \cdot 10^{308})$
bool	1 байт	true / false (0..255)
void	0 байт	Пустой тип

3.2. Приставки к типам данных

unsigned	без знака	*char, int*
long	длинное	*int, float*
short	короткое	*int*

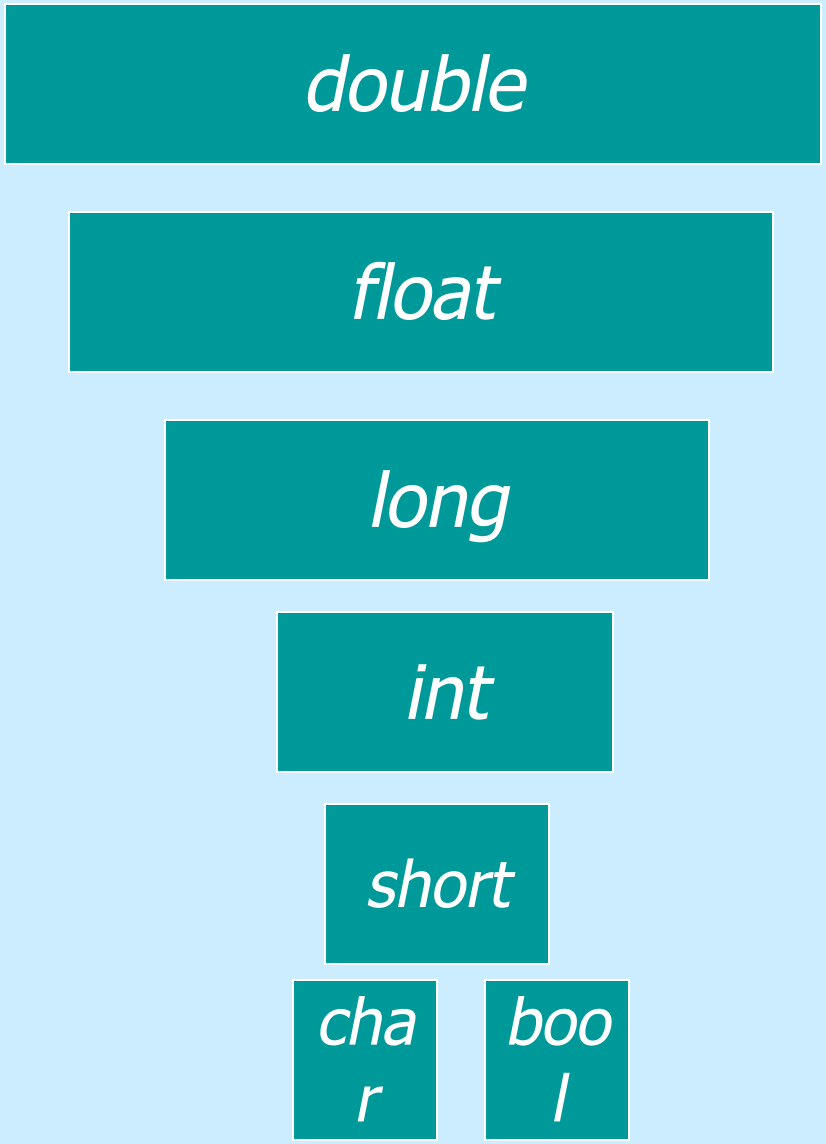
3.3. Преобразование типов

Язык Си поддерживает *неявное преобразование типов.*

```
int x = 5,y = 2,z;
z=x/y;
x = 2.25;
```

```
// z= 2
// x = 2
```

Приоритет типов



Основные правила неявного преобразования типов

Если два операнда выполняемой операции имеют тип *A*, а результат имеет тип *B*, то результат в процессе выполнения операции будет приведен к типу *A*.

```
float z;
z = 1/25;  // переменная z будет равна 0
```

Если два операнда одной операции имеют тип *A* и *B*, а результат имеет тип *B*, то результат будет приведен к типу *B*.

```
int z;
z = 42/2.5;  // переменная z примет значение 16;
...
float z;
z = 42/2.5;  // переменная z примет 16.7666;
```

Если операция выполняется с двумя операндами разных типов, то обе величины приводятся к высшему (по рангу) из типов.

```
int z = 5;
float y= 2.11
z = z/y; // переменная z примет значение 2
```

Явное преобразование типов

```
(char)(120 + 0.5)  /*значение выражения будет приведено к  символу 'x'. */
```

Объявление и инициализация переменных

- от
- оп
  пе
  ис
- не
  ин
- *нельзя дважды использовать при описании один идентификатор*

ых

ых

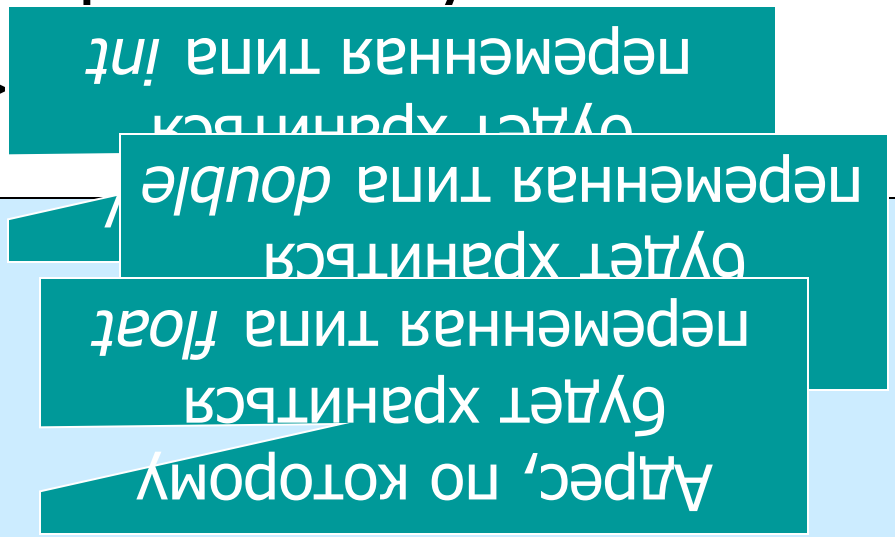
```
int x; // объявление переменной
int k=0; // объявление и
//инициализация переменной
char m = 'c'; // объявление и
//инициализация переменной
x = 13; // инициализация переменной
```

3.4.Производные типы данных

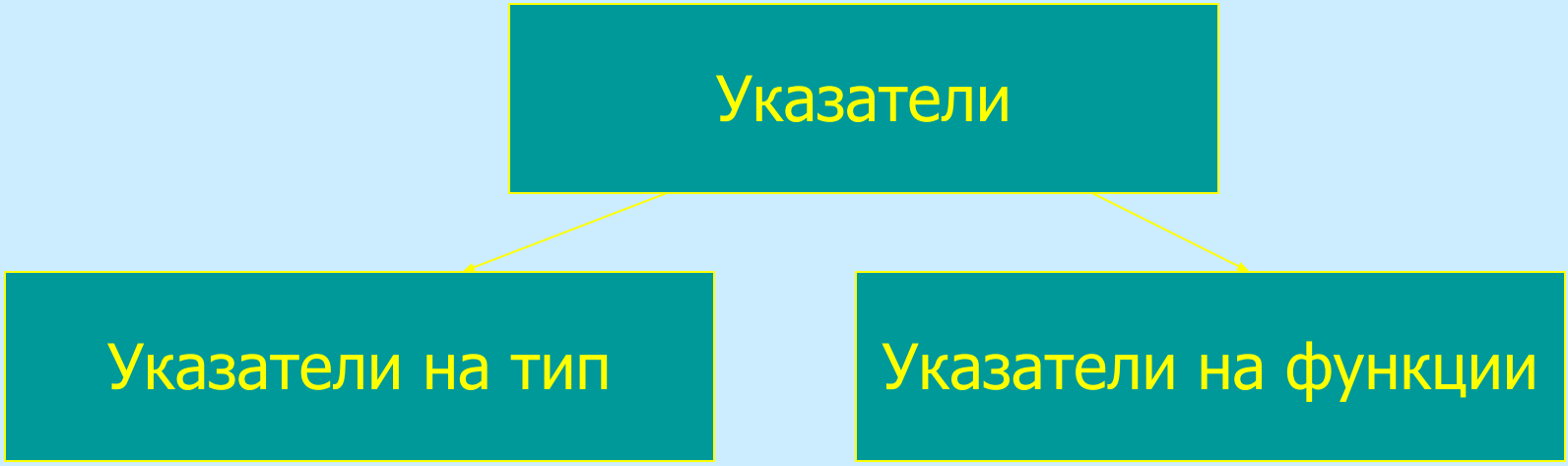
3.4.1. Указатели

Указатель на переменную заданного типа
содержит адрес переменной указанного типа.

Синтаксис: <тип>




```
int *x;
double *y;
float *z;
```



- Занимает в памяти 2 байта
- Рекомендуется обнулять описанный в программе указатель
- Перед использованием указателя необходимо выделить память

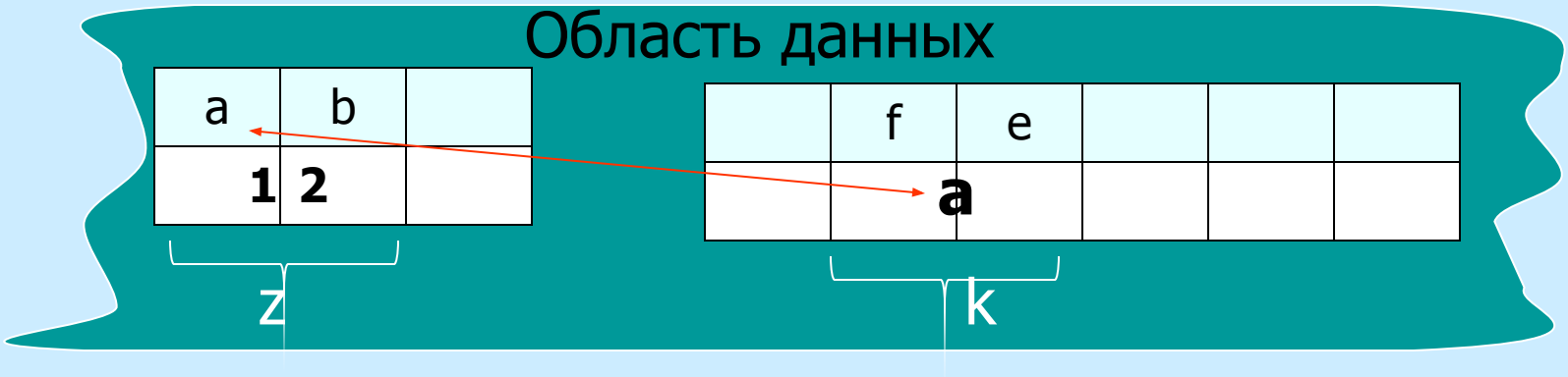
```
// пустой (нулевой) указатель
int  *x = NULL;
```

```
// выделение памяти
x = (int)malloc(sizeof(int));
```

3.4.2. Ссылки

Ссылка – это адрес существующей переменной. Ссылка формируется добавлением знака «&» к имени переменной слева.

```
int z = 12; //объявлена и задана целая переменная
int *k = &z; // указателю k присваивается значение
//адреса переменной z.
```



3.4.3. Разыменование указателей

Для получения или инициализации значения, хранящегося по заданному адресу, используют операцию разыменования указателя - *

```
int z = 12;
int *k = (int)malloc(sizeof(int));
*k = z;
```

4. Конструкции структурного программирования в Си

4.1. Ветвление

Оператор проверки условия if [else]

Синтаксис:

if (логическое выражение)

{действия при истинном значении выражения}

[else {действия при ложном значении выражения}]

Пример

```
…

…

int  m = 12, n = 18;

if  (m<n)

        printf (“Сумма чисел %d”, m+n);

        else printf (“Произведение чисел %d”, m*n);

…

…
```

Пример сложного условия

```
…
int  x = 5, y = 7, z = 3;
int min;
if (x<y&&x<z)
                min = x;
    else if (y<x&&y<z)
                        min = y;
                                else min = z;
…
```