Типы данных. Операции. Выражения Ввод-вывод.

Алфавит языка С

- Буквы латинского алфавита
- Арабские цифры 0 до 9
- Специальные символы :#,$%^&*()!~ /\ и т.д

Лексемы

Минимальные значимые единицы текста в программе

- Идентификаторы
- Ключевые слова
- Знаки операций
- Константы
- Разделители

Идентификаторы (ID)

- называется последовательность цифр и букв, а также специальных символов, при условии, что первой стоит буква или специальный символ.
- Или имя программного объекта – переменной, метки, типа, функции, модуля и тд

Общепринятые правила

- Id переменной строчными буквами – name
- Id типа или функции начинается с заглавной -Name
- Id константы заглавные NAME
- Должен иметь смысловое значение
- Если состоит из нескольких слов, то через _ или новое слово с большой буквы

my_name MyName

- В С строчные и прописные различные символы

Name, NAME, name

- Идентификаторs не могут быть ключевыми словами (~~int~~)

Константы

- *Константы*, являются фиксированными значениями, которые можно вводить и использовать на языках C/C++.
- целые константы,
- константы с плавающей запятой,
- символьные константы
- строковые литералы

```
200             // Целое стандартной размерности
1000000         // Длинная константа
200l, 200L, 0xB8L    // Длинные константы
123u, 60000U     // Беззнаковые константы
077777777UL     // Длинная беззнаковая константа
```

Десятичная константа	Восьмеричная константа	Шестнадцатеричная константа
16	020	0x10
27	0117	0x2B
240	0360	0XF0

Диапазон

$(-2^{31} \ldots +(2^{31} - 1))$ $(0 \ldots 2^{32} - 1)$.

- Символьная константа– представляется символом заключенном в апострофы: ' ', 'Q', '\n', '\\'.

L'a` wchar_t L"asdf".

- Строковая константа (литерал) – последовательность символов кода ASCII (включая строковые и прописные буквы русского и латинского а также цифры) заключенные в кавычки ("): "город Тамбов", “hello".

Ключевые слова

- зарезервированные идентификаторы, которые наделены определенным смыслом

auto double int struct break else long switch
register typedef char extern return void case float
unsigned default for signed union do if sizeof
volatile continue enum short while _asm fortran
near
far cdecl huge paskal interrupt

Комментарии

- набор символов, которые игнорируются компилятором.

```
/*Эта программа выводит сообщение на экран*/

//Эта программа выводит сообщение на экран
```

Структура программы

#директивы препроцессора
. . . . . .
#директивы препроцессора
функция а ( )
    операторы
Определение прототипов функций
Определение глобальных переменных
void main ( ) //функция, с которой начинается выполнение программы
    операторы
      описания
      присваивания
      вызов функции
      пустой оператор
          составной
          выбора
          циклов
          перехода
функция в ( )
    операторы

Препроцессорные директивы

- Начинаются с символа #
- 1. Подключение заголовочных файлов

Формат:

#include <ID_файла>

<> - поиск в стандартной директории

“” – поиск в текущей директории

stdio

iostream cmath

- 2. Обработка макроопределений

- #define <ID> <строка>
- #define SIZE 100

Функция main

- Управление всей работой проекта

```
int _tmain(int argc, _TCHAR* argv[])
{

}
```

stdafx.h stdafx.cpp

Переменная

- переменная– это символическое обозначение ячейки оперативной памяти программы, в которой хранятся данные

Декларация объектов

- Объявление - описание не приводящее к выделению памяти

<класс памяти> <тип> <список объектов>

- Определение , при кот для объекта выделяется объем памяти и его можно инициализировать – задать начальное значение

тип имя_переменной = начальное_значение;

Время существования и область видимости переменных

Областью действия

– блок операторов ({...});

– модуль (файл);

– вся программа в целом.

Временем жизни

-Локальное

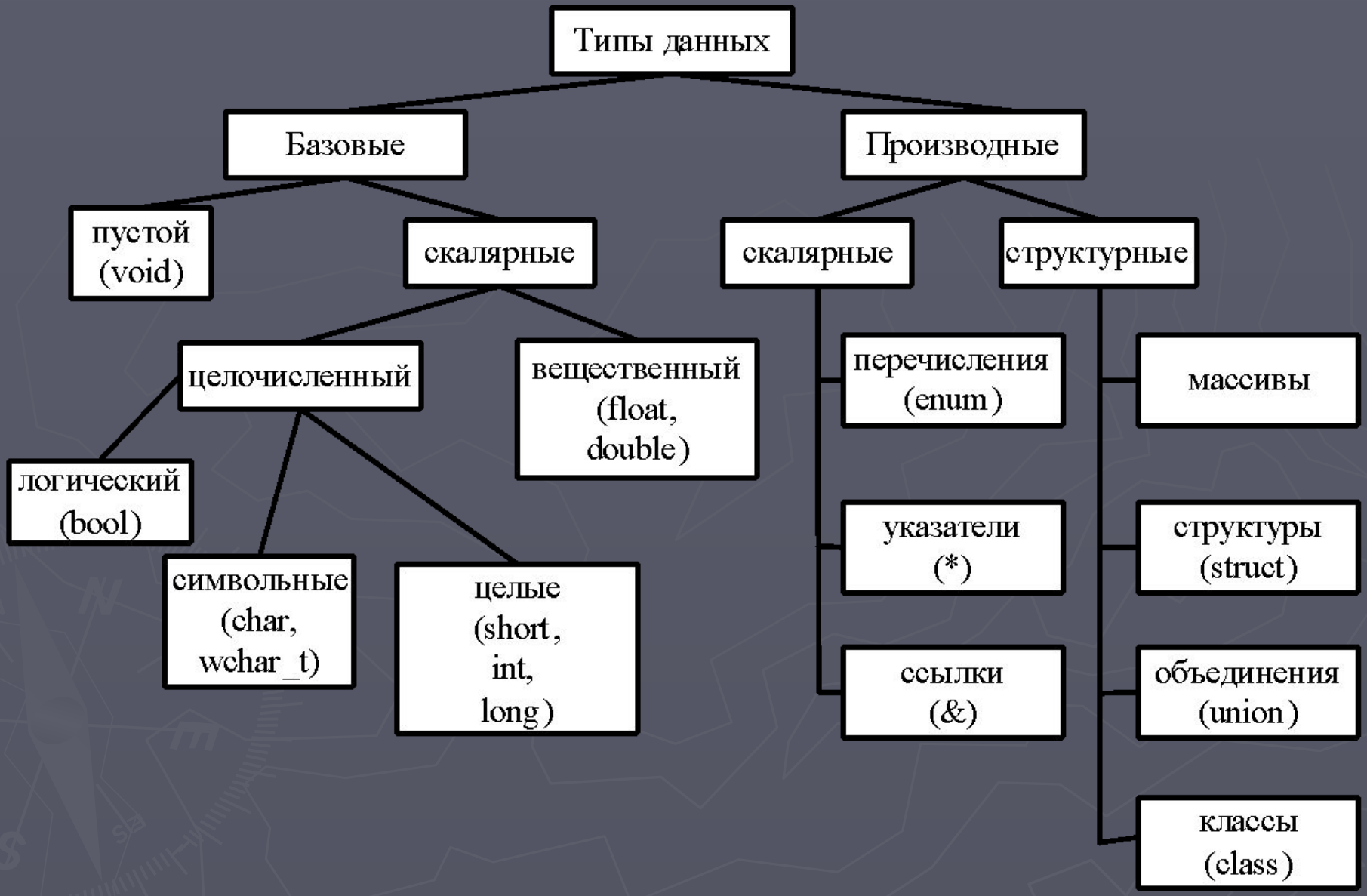
-глобальное

<класс памяти>

- Способ размещение объекта в памяти определяет область видимости и время жизни переменной
- По умолчанию auto

Спецификаторы класса памяти

- auto
- static
- extern
- register


Типы данных
Базовые
Производные
пустой (void)
скалярные
скалярные
структурные
целочисленный
вещественный (float, double)
перечисления (enum)
массивы
логический (bool)
указатели (*)
структуры (struct)
символьные (char, wchar_t)
целые (short, int, long)
ссылки (&)
объединения (union)
классы (class)

<limits.h>

Базовые типы данных целых чисел

- int // целое со знаком, слово
- char // целое со знаком, байт
- unsigned char // целое без знака, байт
- unsigned // целое без знака, слово
- long // целое со знаком, двойное слово
- unsigned long // целое без знака, двойное слово
- short // целое со знаком, короткое слово
- // (слово или байт)

Числа с плавающей точкой

$X = m * p^2$, например $25.4 = 0.254 * 10$

обычной (**float**), - 4 байта

$(\pm 3{,}4 \cdot 10^{-38} \ldots \pm 3{,}4 \cdot 10^{+38})$

двойной (**double**) - 8 байт

$(\pm 1{,}7 \cdot 10^{-308} \ldots \pm 1{,}7 \cdot 10^{+308}$

повышенной (**long double**) точн -10 байт

$(\pm 3{,}4 \cdot 10^{-4932} \ldots \pm 3{,}4 \cdot 10^{+4932})$

E,e -для **double,**
F,f -для **float,**
L,l -для **long double**
.
0.4 .665 3.1415926 1.17e2 -.176E-3
1.1F 3.33L

- [цифры].[цифры] [E|e [+|-] цифры]

Представление символьных данных

- **char**
- unsigned char

Расширенный символьный тип (wchar_t)

Wide CHaracter Type

ASCII – American Standart Code for Information Interchange

```
wchar_t  letter = L'D';  // Unicode
```

```
char  r;
long t;
int i,j,k;
int m=8;
char let=‘s’;
```

```
sizeof (int) ;
// размерность типа данных int  - 4
long   l;
Sizeof( l) ;
   // размерность переменной типа long - 4
sizeof(i+2.0);
// размерность значения выражения типа double - 8
```

Операции. Классификация операций

- *унарные* (воздействуют на одно значение или выражение),
- *бинарные* (участвуют два выражения)
- *тернарных* (три выражения).

- -арифметические **(+,-,*,/,%);**
- -логические **(&&, | |, !);**
- -сравнения **(<,>,>=,<=,==,!=);**
- -машинно-ориентированные (операции над машинными словами, поразрядные **(&,|,^,~,<<,>>);**

- -присваивание **(=,+=,-=,*-,/= и т.д.);**
- -работа с указателями и памятью **(*,&,sizeof);**
- -выделение составляющего типа данных **((),*,[], . , ->);**
- -явное преобразование типа **((тип));**
- -условная **(?:);**
- -последовательность выражений **(","-запятая).**

Приоритеты операций

Ранг	Операции
1	( ) [ ] –> .
2	! ~ – ++ -- & * (тип) sizeof тип( )
3	* / % (мультипликативные бинарные)
4	+ - (аддитивные бинарные)
5	<< >> (поразрядного сдвига)
6	< > <= >= (отношения)
7	== != (отношения)
8	& (поразрядная конъюнкция «И»)
9	^ (поразрядное исключающее «ИЛИ»)
10	| (поразрядная дизъюнкция «ИЛИ»)
11	&& (конъюнкция «И»)
12	|| (дизъюнкция «ИЛИ»)
13	?: (условная операция)
14	= *= /= %= -= &= ^= |= <<= >>= (операция присваивания)
15	, (операция запятая)

Арифметические операции

- + - * /
- Операнды- константы, переменные, функции, эл. массивов, указатели и арифмет. выражения

Порядок выполнения

1) Выражения в круглых скобках
2) Функции (стандартные мат., пользовательские)
3) * / %
4) - +

```
a = (a + 1) % 16;
// a присвоить a+1 по модулю 16
```

Операции присваивания

- -обычное присваивание **(=);**
- -присваивание, соединенное с одной их бинарных операций **или составное присваивание (+=, -=, *=, /=, %=, <<=, >>=, &=, |=, ^=);**
- -операции инкремента и декремента (увеличения и уменьшения на 1).

```
long a; char b; int c;
a = b = c;    // эквивалентно b = c; a = b;


a +=b; // эквивалентно a = a + b;

x=i+(y=3)-(z=0); //z=0  y=3  x=i+y-z
```

недопустимо

- Присвоение константе

2=a+f;

- Присвоение функции

getch()=i;

- Присвоение результату операции

(i+1)=2+y;

ИНКРЕМЕНТ И ДЕКРИМЕНТ

```
int a;
// Эквивалент    Интерпретация
a++;
// Rez=a; a=a+1; Увеличить на 1 после использования
++a;
// a=a+1; Rez=a; Увеличить на 1 до использования
a--;
// Rez=a; a=a-1; Уменьшить на 1 после использования
--a;
// a=a-1; Rez=a; Уменьшить на 1 до использования
++(x + 1); // ошибка
```

Примеры

```
int i, j,  k;
float x,  y;
   x*=y;        // x=x*y;
   i+=2;        //i=i+2;
   x/=y+15;  //x=x/(y+15);
   --k;       //k=k-1;
   j=i++;      //j=i; i=i+1;
   j=++i;      //i=i+1; j=i;
```

Операции сравнения и логические операции

== != <= >= < >

<выражение1> <знак> <выражение2>

правила

1.опреранды – любые базовые типы

2. выражения перед сравнением преобразуются к одному типу

3. результат 1- если отношение истинно и 0- если ложно

```
a = b > c;
   // Запомнить результат сравнения
   a = (b > c)* 2
   // Принимает значения 0 или 2
```

ЛОГИЧЕСКИЕ ОПЕРАЦИИ

- **НЕ (!)**
- **И (&&) конъюнкция**
- **ИЛИ (| |) дизъюнкция**
-

```
if (a < b && b < c)
// если ОДНОВРЕМЕННО ОБА  a < b и b < c,
   то...

if (a==0 || b > 0)
// если ХОТЯ БЫ ОДИН a==0 или b > 0,
   то...
```

```
while(!k) {...}
// эквивалентно  while(k==0) {...}

!0              // 1
!5            //0
!((x=10)>0)    //0
```

Условная операция

```
int a;
   double b;
   c = x + a > b ? a : b;


// Условие ? Выражение для "истина" :
   Выражение для "ложь"
```

Операция последовательности действий ("запятая")

выражение 1, …., выражение M

m=(i=1,j=i++,k=6,n=i+j+k)

// i=1, j=i=1, i=2,k=6, n=2+1+6, m=n=9

Роль символа ";"

a = b + c – 5

if (a < b) else

Выражение + “;” -- > оператор

Побитовые логические операции

- ~ - дополнение , инвертирование
- & - побитовое И - конъюнкция
- | - побитовое включающее ИЛИ –дизъюнк.
- ^ - побитовое исключающее ИЛИ – сложение по модулю 2
- >> сдвиг вправо
- << сдвиг в лево

- нельзя применять к переменным вещественного типа

~

~0 1

~1 0

&

1	1	0	0		
1	0	1	0	___	___
1	0	0	0		

1	1	0	0				
1	0	1	0				
1	1	1	0				

^

1	1	0	0		
1	0	1	0		
0	1	1	0		

```
~0xF0      //0x0F
0xFF & 0x0F //0x0F
0xF0 | 0x11  //0xF1
0xF4^0xF5   //0x01
     11110100
     11110101
     --------
     00000001
```

x=1
y=2

x & y //0 - 0001 & 0010=0000

СДВИГ

0x81<<1 //10000001<<1 =00000010 =0x02

0x81>>1 //10000001>>1=01000000=0x40

правила

1. если тип unsigned то заполняются нулями
2. если signed то результат не определен

Применение

сдвиг вправо на k разрядов – деление на 2 в степени k

```
x>>1        //x/2
```

сдвиг влево на k разрядов – умножение на 2 в степени k

```
x<<1        // x*2
x<<3        //x*8
```

Применение

- маскирование разрядов

проверка нечетности целого

```
int i;


if (i&1)  printf (“Значение i нечетно”);
```

Неявное преобразование

- short, int → unsigned long → double
- char → float → double

```
float x;  int i;

x+i ;

// int в float
```

Явное преобразование типа

(тип) выражение;

```
double x,d;          // double x,d; int n;
d = x - (int)x;      // n = x; d = x - n;
```

Пример

```
float a;
int i = 6, j = 4;

a = (i + j) / 3;              // a=3
a = (float)(i + j) / 3;  →  // a=3.333333
```

Ошибки

```
if (a=b)  //if (a==b)
..(a<<3)  //… (a<3)
if (a &&  0x10)  // if (a &0x10)

char  c[80];
#define CODE 1
if (c[i]==CODE)  //(int)c[i]==1

0 < x < 100        //ошибка
(0 < x) && (x < 100)//верно
```

```
int a, b;  long   c;
c=a*b;  // некорр   2 147 483 647
c=(long)a*b; //правильно
```

Стандартные мататематич функции

cmath

stdlib.h

большинство тип double

аргументы тригонометрических – радианы

$\sqrt{x}$ sqrt(x)

|x| fabs(x)

e^x	exp(x)
x^y	pow(x,y)
ln(x)	log(x)
lg10(x)	log10(x)
tg(x)	tan(x)
остаток от деления	fmod(x,y)
наименьшее целое	ceil(x)
наибольшее целое	floor(x)

```
z=pow(x,10.)+3.7*pow(x,8.);
```

$z=x^{10}+3.7x^{8}$