Семинар 1.
Лексические основы, арифметические типы данных, переменные и константы, операторы, линейный алгоритм

2. Арифметические типы данных, переменные и константы

1. Что такое переменная
2. Классификация типов данных
3. Определение и описание переменных
4. Ключевые слова для определения и описания переменных (типы данных)
5. Константы: препроцессорные и именованные

2.1. Что такое переменная

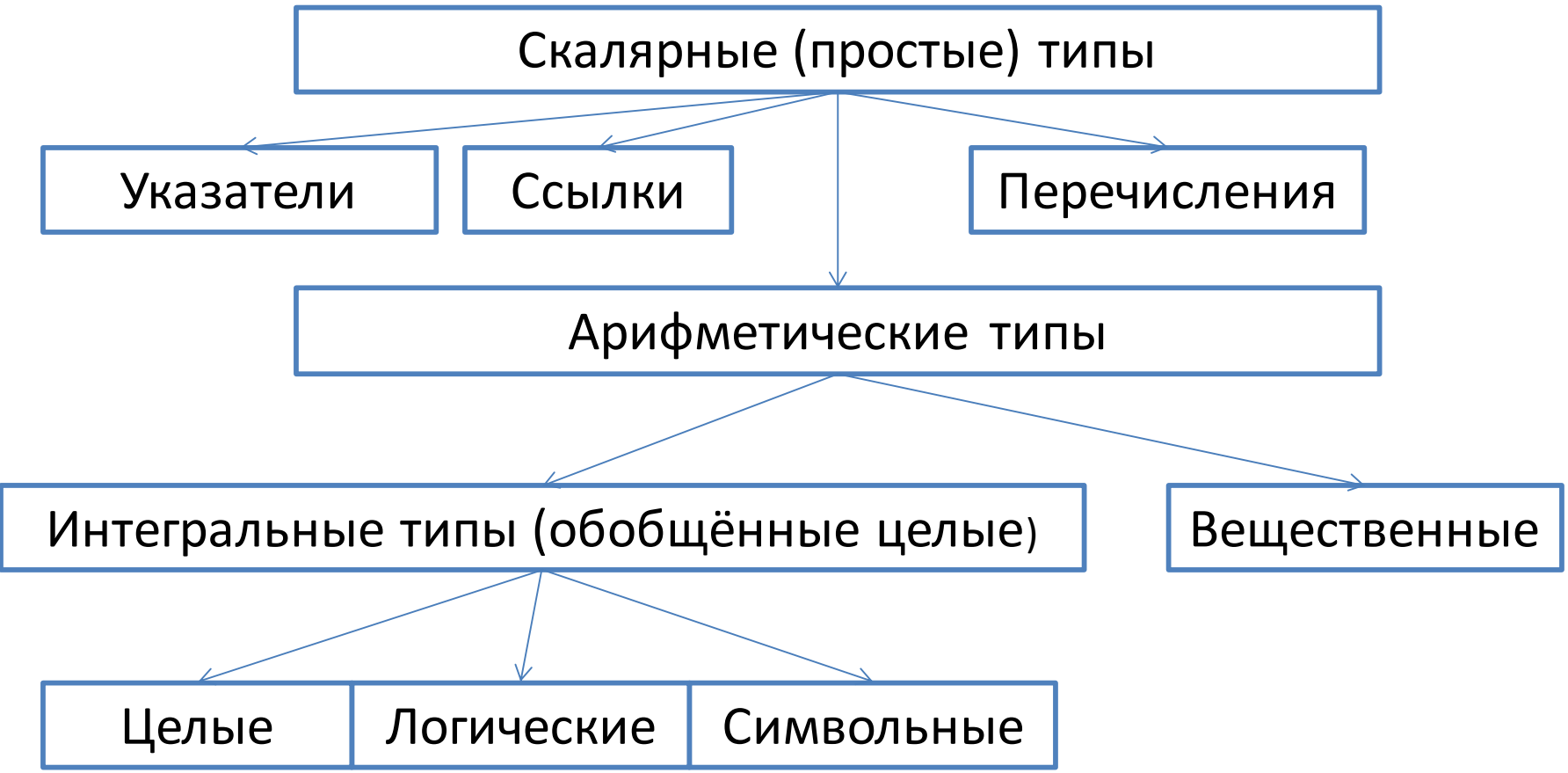
В языке C++ **переменная – частный случай объекта**; объект – непрерывный участок памяти, для которого **типом объекта определены:**

- **размеры,**
- **структура,**
- **множество возможных значений,**
- **набор допустимых операций**

2.1. Что такое переменная

Переменную чаще всего определяют как пару «имя-значение». Имени соответствует адрес участка памяти, выделенный переменной, а значением является содержимое этого участка.

2.2. Классификация типов данных



2.2. Классификация типов данных

Все типы делятся на:

1) скалярные (простые)
2) структруированные (составные)

Примеры структурированных типов:

- строка
- массив
- класс, структура

2.3. Определение и описание переменных

Формат определения: тип имя [= значение]; или тип имя [(значение)];

Пример:

- int i = 2; // определение
- extern int j; // описание
- char letter = 's';
- bool isEmpty = false;
- float number1 = 3.67;

2.3. Определение и описание переменных

Описание является определением, если:

- вводит переменную
- содержит инициализатор
- полностью описывает функцию (включает тело функции)
- вводит объединение или структуру
- вводит класс
- вводит шаблон классов или функций

2.3. Определение и описание переменных

Описание не является определением, если:

- это прототип функции
- содержит спецификатор extern
- вводит статический компонент класса
- вводит имя класса
- вводит имя типа (typedef)
- вводит прототип шаблона

2.4. Базовые типы данных

Имя типа	Размер, байт	Диапазон (см. limits)
bool	1	{true, false}
char	1	[CHAR_MIN; CHAR_MAX]
short	2	[SHRT_MIN; SHRT_MAX]
int	4	[INT_MIN; INT_MAX]
long	4	[LONG_MIN; LONG_MAX]
float	4	[-2 147 483 648.0; 2 147 483 647.0]
double	8	[-9 223 372 036 854 775 808 .0; 9 223 372 036 854 775 807.0]

2.4. Базовые типы данных

Пример. Операция sizeof.

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int _tmain()
{
  cout << "sizeof(bool) = " << sizeof(bool) << " byte(s)" << endl;
  cout << "sizeof(int) = " << sizeof(int) << " byte(s)" << endl;
  cout << "sizeof(long) = " << sizeof(long) << " byte(s)" << endl;
  return 0;
}
```

2.4. Базовые типы данных

В обозначении типа может использоваться одновременно несколько служебных слов:

```
long double ld1 = 1.12345678901234567890;
/* вещественный тип расширенной точности */
```

2.4. Базовые типы данных

Служебные слова unsigned и signed позволяют выбрать способ учёта знакового разряда:

- unsigned = «без знака»
- signed = «со знаком»

Пример: unsigned int ui1;
// ui1 может принимать целые положительные значения от 0 до UINT_MAX

2.4. Базовые типы данных

Служебное слово signed обычно опускается:

signed int	*эквивалентно*	int
signed char		char
signed short		short
signed long		long
unsigned		unsigned int

2.4. Базовые типы данных

Длинные целые числа можно разделять апострофами:

```
int x = 2'000'000;
cout << x << endl;
```

2.4. Базовые типы данных

Типы из файла cstdint

#include <cstdint>:

```
int8_t i8; // Целое со знаком, 8 бит
uint8_t ui8; // Целое без знака, 8 бит
int16_t i16; // Целое со знаком, 16 бит
uint16_t ui16; // Целое без знака, 16 бит
int32_t i32; // Целое со знаком, 32 бит
uint32_t ui32; // Целое без знака, 32 бит
int64_t i64; // Целое со знаком, 64 бит
uint64_t ui64; // Целое без знака, 64 бит
```

2.4. Базовые типы данных

Типы из файла cstdint
#include <cstdint>:

```
cout << sizeof(i8) << endl;

#include<limits>
cout << "Minimum of int8_t:" <<
numeric_limits<int16_t>::min() << endl;
cout << "Maximum of int8_t:" <<
numeric_limits<int16_t>::max() << endl;
```

2.4. Базовые типы данных

Б. Страуструп: «В большинстве приложений можно обойтись типами bool для логических значений, int – для целых, char – для символов и double – для чисел с плавающей точкой. Остальные фундаментальные типы являются вариациями для оптимизации и решения других специальных задач.»

2.4. Базовые типы данных

Спецификатор typedef позволяет вводить удобные названия для сложных обозначений типов:

```
typedef unsigned char uch;
uch mySymbol;
```

2.4. Базовые типы данных

Из базовых типов данных можно конструировать множество производных типов:

- массивы
- функции
- указатели
- ссылки
- классы
- перечисления

2.4. Базовые типы данных

Пример. Для каждого перечисления может быть введено имя типа:

- enum week {sunday, monday, tuesday, wednesday, thursday, friday, saturday};

enum – служебное слово, позволяющее создать перечисление
week – имя созданного производного типа

2.5. Константы: препроцессорные и именованные

Препроцессорная константа:

#define идентификатор строка_замещения

Примеры:

```
#define begin {
#define K 30
#define HW "Hello world"
```

2.5. Константы: препроцессорные и именованные

Препроцессорные замены происходят до компиляции программы и не выполняются внутри строковых и символьных констант и комментариев

2.5. Константы: препроцессорные и именованные

Именованные константы:

const тип имя_константы инициализатор

Пример:

```
const int five = 5;
const int six(6);
```

2.5. Константы: препроцессорные и именованные

Пример. Использование констант.

```
#include "stdafx.h"
#include <iostream>
using namespace std;
#define begin {
int _tmain()
begin
  enum {zero, one, two, dva = 2, three};
  const int nn = 40;
  cout << "nn = " << nn << endl;
  cout << "one = " << one << endl;
  return 0;
}
```