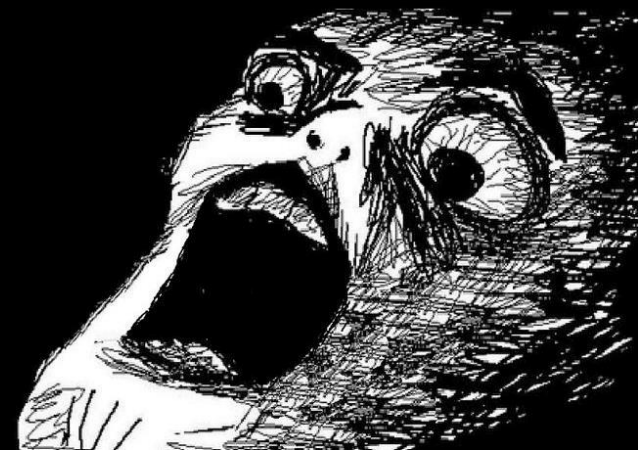
Адресация и указатели

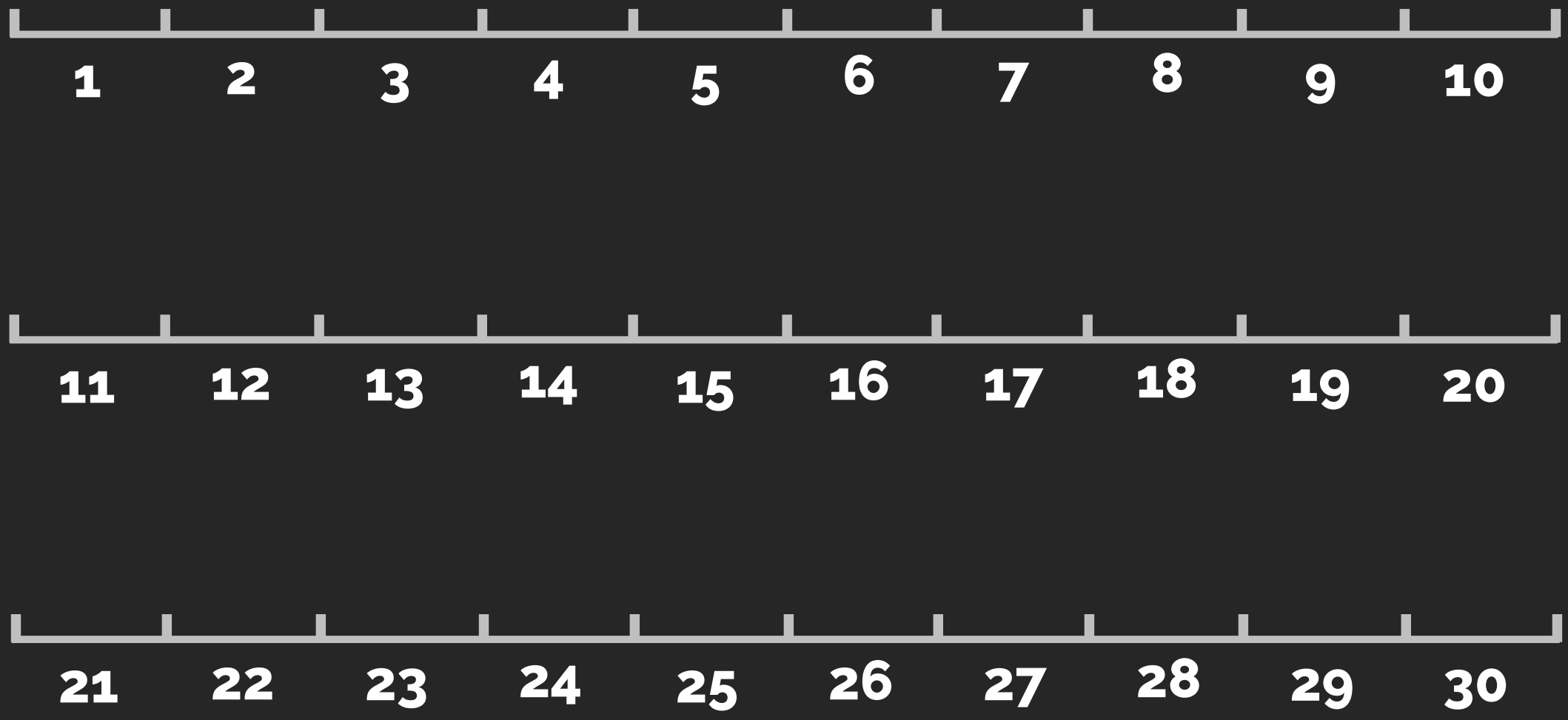
УКАЗАТЕЛИ

УКАЗАТЕЛИ



Все переменные хранятся в оперативной памяти

Оперативная память – это большой склад


1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30

1 2 3 4 5 6 7 8 9 10

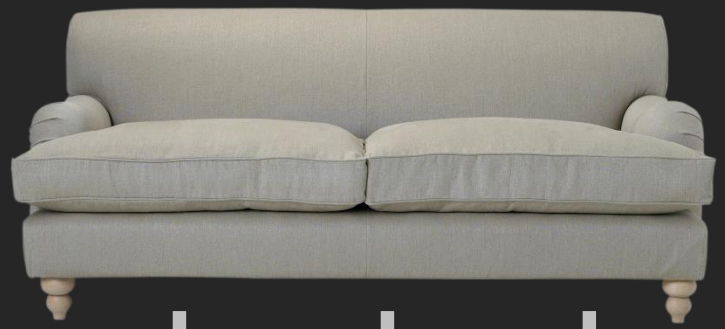
11 12 13 14 15 16 17 18 19 20

21 22 23 24 25 26 27 28 29 30

1 2 3 4 5 6 7 8 9 10

11 12 13 14 15 16 17 18 19 20

21 22 23 24 25 26 27 28 29 30



1 2 3 4 5 6 7 8 9 10

11 12 13 14 15 16 17 18 19 20

21 22 23 24 25 26 27 28 29 30

Каждый предмет имеет номер (адрес)
Каждый предмет имеет свой размер

С журналом можно не помнить, где хранятся предметы

Полка 1 – Яблоки (3 шт)
Полка 2 – Вода (7,66 л)
Полка 4 – Диван
...

```
int a;
// Выделить в памяти (на складе) ячейку для целых чисел.
// Номер ячейки связать с именем a.
```

```
int a;
a = 5;
// Поместить в ячейку a значение 5
// (программа сама найдет ячейку с нужным адресом)
```

```
int a;
a = 5;
a = a + 2;
// Добавить к переменной a еще 2
```

```
int a;
a = 5;
a = a + 2;
cout << a;
// Получить значение переменной из памяти и вывести его на экран
```

```
int a;
a = 5;
a = a + 2;
cout << a;
cout << &a;
// Получить адрес переменной a и вывести его на экран
```

& - операция взятия адреса (пишется слева от имени переменной)

```
int a;
a = 5;
a = a + 2;
cout << a;
cout << &a;
cout << sizeof(a);
// Получить размер переменной a и вывести его на экран
```

sizeof – функция, возвращающая размер переменной

```
int a;
a = 5;
a = a + 2;
cout << a;
cout << &a;
cout << sizeof(a);
```

```
7
001200AF
4
```

Каждая переменная имеет адрес
Каждая переменная имеет свой размер
Каждая переменная имеет имя
Каждая переменная имеет значение

```
int a;
a = 5;
a = a + 2;
cout << a;
cout << &a;
cout << sizeof(a);

int b;
b = a + 5;
cout << b;
cout << &b;
cout << sizeof(b);
```

```
7
001200A0
4

12
001200A4
4
```

Переменные одного типа имеют одинаковый размер

```
int a;
a = 5;
a = a + 2;
cout << a;
cout << &a;
cout << sizeof(a);

double b;
b = 7.46;
cout << b;
cout << &b;
cout << sizeof(b);
```

```
7
0010AABA
4

7.46
0010AABD
8
```

Переменные разных типов имеют разный размер

Если переменная занимает несколько ячеек, то какой у неё адрес?

```
int main()
{
   int a = 5;
   int& b = a;

   cout << "Address of a: " << &a << endl;
   cout << "Address of b: " << &b << endl;

   cout << endl;
   b = 7;
   cout << "Value of a: " << a << endl;
}
```

**Ссылка – это переменная,
которая хранит адрес другой переменной**

Синтаксис:
type& reference = name;
Пример:
int& b = a;

**После инициализации ссылки,
адрес в ссылке изменить нельзя**

```
int main()
{
   int a = 5;
   int& b = a;

   cout << "Address of a: " << &a << endl;
   cout << "Address of b: " << &b << endl;

   cout << endl;
   b = 7;
   cout << "Value of a: " << a << endl;
}
```

```
Address of a: 00AFFDFC
Address of b: 00AFFDFC

Value of a: 7
```

```cpp
int main()
{
    int a = 5;
    int& b = a;

    cout << "Address of a: " << &a << endl;
    cout << "Address of b: " << &b << endl;

    cout << endl;
    b = 7;
    cout << "Value of a: " << a << endl;
}
```

```
Address of a: 00AFFDFC
Address of b: 00AFFDFC

Value of a: 7
```

Различайте использование &

- Когда & после типа данных, это ссылка
- Когда & без типа перед именем переменной,
  это взятие адреса
- Когда & между двумя логическими переменными,
  это логическое умножение

```
double& ref = d;
cout << &d;
bool one; bool two;
bool three = one & two;
```

Если ссылка ссылается на тот же адрес, что и переменная, почему не использовать саму переменную?

```
int main()
{
   double a = 5.0;
   cout << "Address of a in main(): " << &a << endl;
   cout << "Value of a in main(): " << a << endl;
   cout << endl;

   Foo(a);

   cout << endl;
   cout << "Value of a in main(): " << a << endl;
}
```

```
void Foo(double a)
{
   cout << "Address of a in Foo(): " << &a << endl;
   cout << "Value of a in Foo(): " << a << endl;

   a = 15.0;
   cout << "New value of a in Foo(): " << a << endl;
}
```

```
int main()
{
   double a = 5.0;
   cout << "Address of a in main(): " << &a << endl;
   cout << "Value of a in main(): " << a << endl;
   cout << endl;

   Foo(a);

   cout << endl;
   cout << "Value of a in main(): " << a << endl;
}
```

```
void Foo(double a)
{
   cout << "Address of a in Foo(): " << &a << endl;
   cout << "Value of a in Foo(): " << a << endl;

   a = 15.0;
   cout << "New value of a in Foo(): " << a << endl;
}
```

```
Address of a in main(): 00D3F814
Value of a in main(): 5

Address of a in Foo(): 00D3F73C
Value of a in Foo(): 5
New value of a in Foo(): 15

Value of a in main(): 5
```

```
int main()
{
 double a = 5.0;
 cout << "Address of a in main(): " << &a << endl;
 cout << "Value of a in main(): " << a << endl;
 cout << endl;

 Foo(a);

 cout << endl;
 cout << "Value of a in main(): " << a << endl;
}
```

```
void Foo(double& a)
{
 cout << "Address of a in Foo(): " << &a << endl;
 cout << "Value of a in Foo(): " << a << endl;

 a = 15.0;
 cout << "New value of a in Foo(): " << a << endl;
}
```

```
int main()
{
 double a = 5.0;
 cout << "Address of a in main(): " << &a << endl;
 cout << "Value of a in main(): " << a << endl;
 cout << endl;

 Foo(a);

 cout << endl;
 cout << "Value of a in main(): " << a << endl;
}
```

```
void Foo(double& a)
{
 cout << "Address of a in Foo(): " << &a << endl;
 cout << "Value of a in Foo(): " << a << endl;

 a = 15.0;
 cout << "New value of a in Foo(): " << a << endl;
}
```

```
Address of a in main(): 00D3F814
Value of a in main(): 5

Address of a in Foo(): 00D3F814
Value of a in Foo(): 5
New value of a in Foo(): 15

Value of a in main(): 15
```

Если вы хотите передать только значение, используйте обычные переменные в функциях

Если вы хотите изменить значение в переменной, используйте ссылки

Передача по значению

```
void Foo(double a)
{
   ...
}
```

Передача по ссылке

```
void Foo(double& a)
{
   ...
}
```

Язык Си++ - строго типизированный язык

Тип данных – это множество допустимых значений и операций над ними

Адреса имеют собственные уникальные значения (восемь 16-ых цифр), и мы можем делать над ними операции

Адрес – это тоже тип данных

Для хранения адресов придуманы специальные переменные - указатели

```
int a = 5;
int* pointer = &a;

cout << "Address of a: " << &a << endl;
cout << "Address in pointer: " << pointer << endl;
```

```
Address of a: 00B9FCF4
Address in pointer: 00B9FCF4
```

```
int a = 5;
int* pointer = &a;

cout << "Address of a: " << &a << endl;
cout << "Address in pointer: " << pointer << endl;
cout << "Address of pointer: " << &pointer << endl;
```

```
Address of a: 00B9FCF4
Address in pointer: 00B9FCF4
Address of pointer: 00B9FCE8
```

```
int a = 5;
int* pointer = &a;

cout << "Address of a: " << &a << endl;
cout << "Address in pointer: " << pointer << endl;
cout << "Address of pointer: " << &pointer << endl;

cout << endl;
*pointer = 7;
cout << "Value in a: " << a << endl;
cout << "Value by pointer address: "
                              << *pointer << endl;
```

```
Address of a: 00B9FCF4
Address in pointer: 00B9FCF4
Address of pointer: 00B9FCE8

Value in a: 7
Value by pointer address: 7
```

Указатель – это специальный тип данных, который хранит адреса других переменных

Синтаксис:
type* reference = &name;
Пример:

```
int a = 7;
int* b = &a;
```

В отличие от ссылок, адрес в указателе можно поменять

**Если оперативная память это склад,
тогда указатель – это листочек
в одной из ячеек склада,
на котором записана ячейка
какого-то предмета на складе**

Например, листочек, который лежит в ячейке 10,
на котором написано «3 яблока в ячейке 1»

Различайте использование *

- Когда * после типа данных, это объявление указателя
- Когда * без типа перед именем переменной, это разыменование
- Когда * между двумя численными переменными, это умножение

```
double* ref = d;
cout << *d;
int one; int two;
double three = one * two;
```

Так зачем нужны указатели?

ничего не понимаю

**Указатель всегда занимает 8 байт,
но может хранить адреса
объектов в сотни мегабайт**

**Проще искать/сортировать/
фильтровать/работать с указателями в 8 байт,
чем объекты в сотни мегабайт**

Подробнее по теме:
выполнить лабораторную работу #1

Важно – все примеры лабораторной набирать вручную, а не копировать из методички. Только набирая вручную, вы разберетесь и запомните материал