2. Типы данных, операции, циклы

Программирование и алгоритмизация

Практические занятия

БИВТ-24-17

Надежда Анисимова

ms teams m2102039@edu.misis.ru

Проверка себя

1. Выбрал и настроил под себя IDE/редактор кода
2. Зашел в тг канал практических занятий
3. Зашел в тимс-команду практических занятий

Что запомнили с прошлой пары?

Типы данных

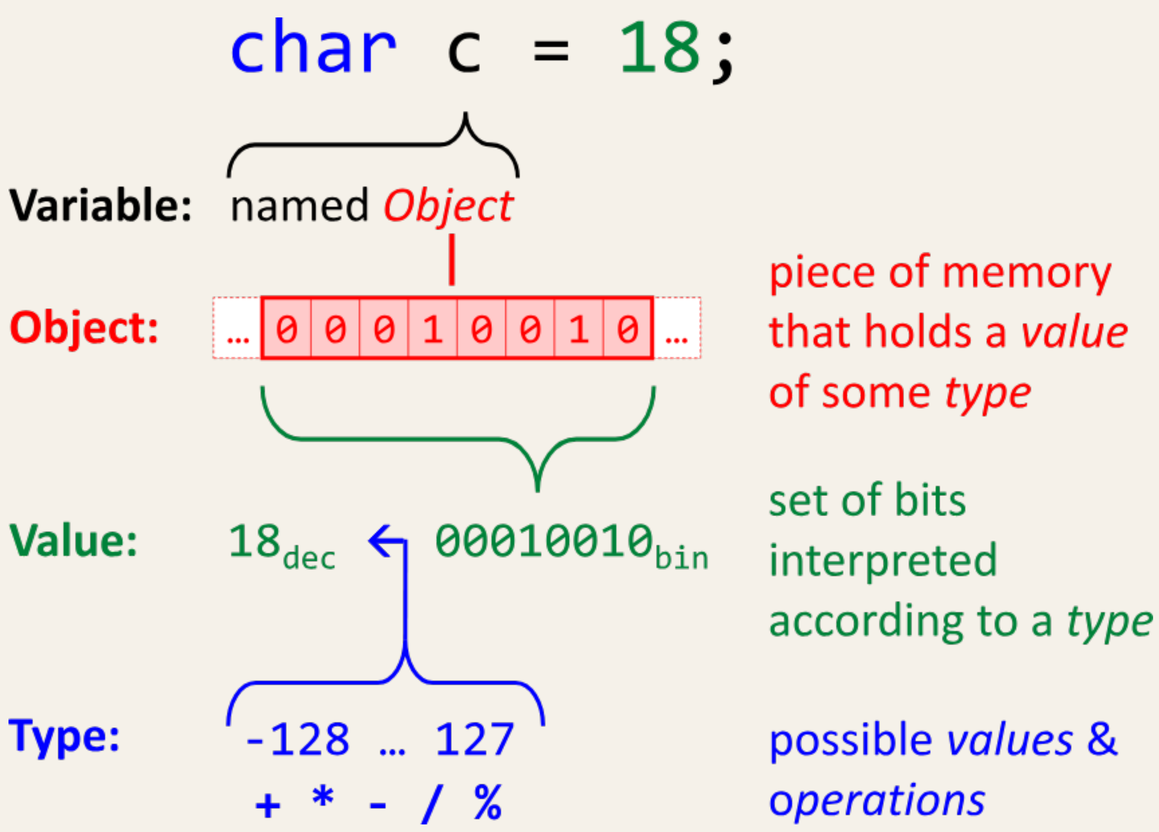
01

Сущности в c++ должны быть определенного типа, чтобы программа скомпилировалась

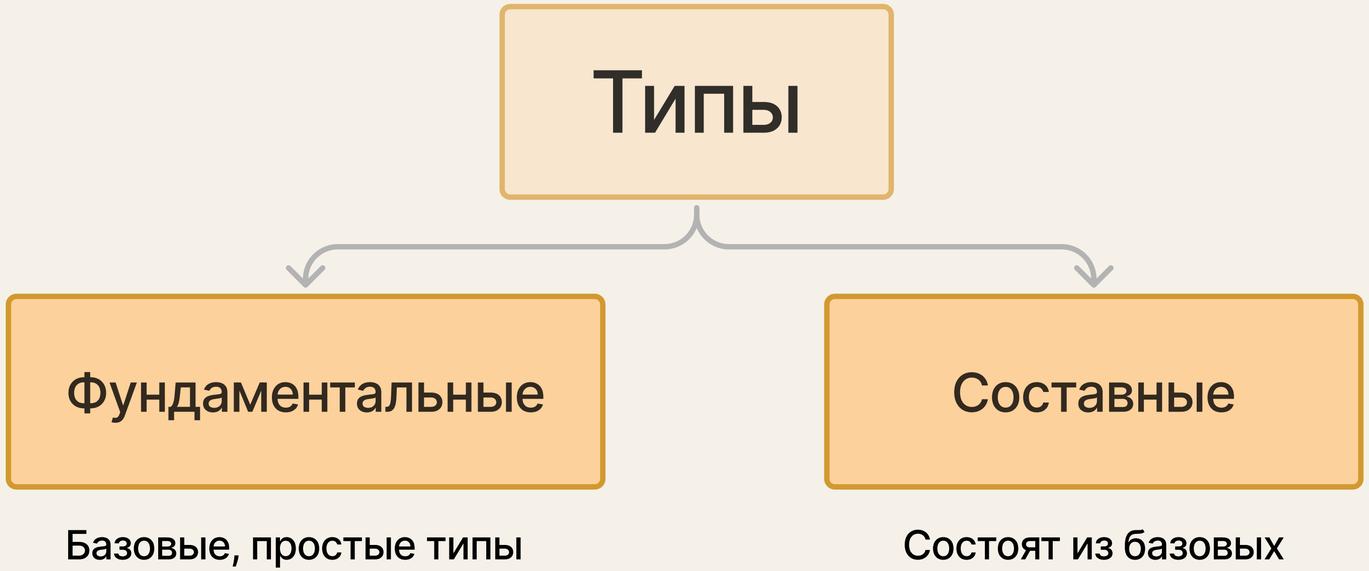
На что указывает тип:

- Сколько нужно выделить памяти под переменную
- Диапазон значений: int от -2^31 до 2^31-1
- Операции, которые можно выполнять с этим типом
- Интерпретация данных в памяти - как интерпретировать биты

Можно создать новый свой тип - класс или структуру



- Кусочек памяти
- Интерпретация битов согласно типу
- Допустимые значения



Фундаментальные:

- Арифметические типы (целочисленные и с плавающей запятой)
- void
- nullptr

Тип	байт	Диапазон принимаемых значений
целочисленный (логический) тип данных		
bool	1	0 / 255 (0 – false, 1-255 – true)
целочисленный (символьный) тип данных		
char	**1**	**0 / 255**
целочисленные типы данных		
short int	2	-32 768 / 32 767
unsigned short int	2	0 / 65 535
int	**4**	**-2 147 483 648 / 2 147 483 647**
unsigned int	4	0 / 4 294 967 295
long int	4	-2 147 483 648 / 2 147 483 647
unsigned long int	4	0 / 4 294 967 295
типы данных с плавающей точкой		
float	**4**	**-2 147 483 648.0 / 2 147 483 647.0**
long float	8	-9 223 372 036 854 775 808 .0 / 9 223 372 036 854 775 807.0
double	**8**	**-9 223 372 036 854 775 808 .0 / 9 223 372 036 854 775 807.0**

По умолчанию типы в c++ не инициализируются!

Zero-overhead principle:

Не нужно платить за то, что не используешь

*желательно типы инициализировать сразу

```
// только объявление
type variable;

// объявление и инициализация
type variable = value;

type variable {value};
```

Переполнение целочисленных типов

```
int x = 2147483647;  // Максимальное значение для int
x++;                 // Переполнение
std::cout << x << std::endl;   // Выведет -2147483648
```

```
unsigned int y = 4294967295;  // Максимальное значение для unsigned int
y = y + 1;                    // Переполнение
unsigned int z = y - 1;  // 4294967295, то есть 2**32 - 1
std::cout << y << " " << z << std::endl;  // Выведет 0 4294967295
```

Операции

02

Арифметические операции

Operator	Name of the Operators	Operation	Implementation
+	Addition	Used in calculating the Addition of two operands	x+y
–	Subtraction	Used in calculating Subtraction of two operands	x-y
*	Multiplication	Used in calculating Multiplication of two operands	x*y
/	Division	Used in calculating Division of two operands	x/y
%	Modulus	Used in calculating Remainder after calculation of two operands	x%y

c = a + b результат записывается в новую переменную **c**

a += b результат записывается в текущую переменную **a**

Инкремент и декремент

операции, которые используются для увеличения или уменьшения значения переменной на единицу

- Префиксный инкремент (++x): Сначала увеличивает значение переменной на 1, а затем возвращает новое значение.

- Постфиксный инкремент (x++): Сначала возвращает текущее значение переменной, а затем увеличивает её на 1

```
int a = 1;
int b = 5;

b = a++; // b = 1, a = 2
b = ++a; // b = a = 3
```

Сравнение

Результат – true или false

```
bool b1 =  x == 5;
bool b2 = (x != 6);
bool b3 = x > y;
bool b4 = x < y;
bool b5 = y >= 5;
bool b6 = x <= 30;
```

Проблемы в сравнении float

числа с плавающей запятой представлены в компьютере в виде двоичной записи с ограниченной точностью

Можно сравнивать с epsilon**, то есть с определенной точностью**

```
float epsilon = 0.00001;

std::abs(a - b) < epsilon;
```

Логические операторы

Операции	Обозначение	Условие	Краткое описание
И	**&&**	a == 3 **&&** b > 4	Составное условие истинно, если истинны оба простых условия
ИЛИ	**||**	a == 3 **||** b > 4	Составное условие истинно, если истинно, хотя бы одно из простых условий
НЕ	**!**	!(a == 3)	Условие истинно, если a не равно 3

1. Логическая операция И && AND;
2. Логическая операция ИЛИ || OR;
3. Логическая операция НЕ ! или логическое отрицание NOT.

Побитовые операторы

1. Bitwise AND (&)
2. Bitwise OR (|)
3. Bitwise XOR (^)
4. Bitwise NOT (~)
5. Left Shift (<<)
6. Right Shift (>>)

```
char a, b = 5, c = 11; // b = 00000101, c = 00001011
a = b & c; // a = 00000001
a = b | c; // a = 00001111
a = ~c;    // a = 11110100
a = b ^ c; // a = 00001110
```

Условия и циклы

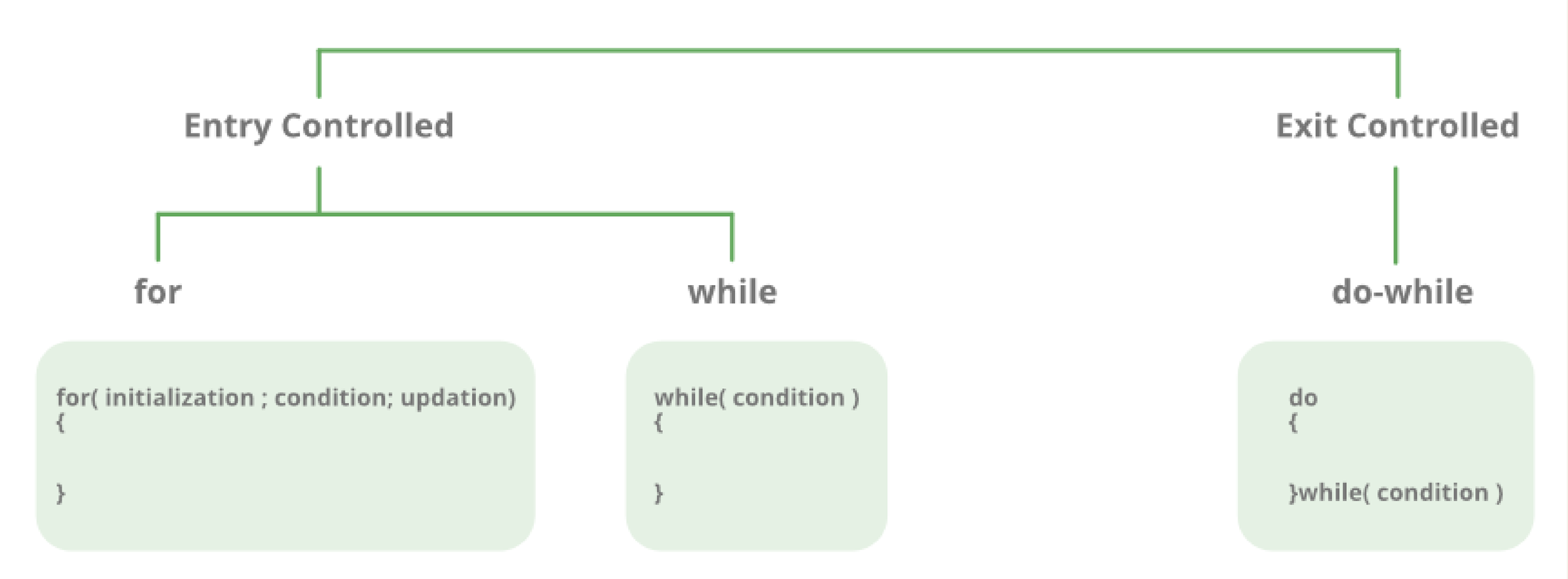
03

Условия

if, else if, else

```
if (condition1) {
    // случай, когда condition1 истинно
} else if (condition2) {
    // случай, когда condition1 ложно, а condition2 истинно
} else if (condition3) {
    // случай, когда condition1 и condition2 ложны, а condition3 истинно
} else {
    // случай, когда condition1, condition2 и condition3 ложны
}
```

Циклы



Операторы break, continue

break – при условии выйти из цикла

continue – при условии пропустить итерацию

- for
- while
- do…while

Циклы

```
for (инициализатор; условие; итерация)
{
   // тело цикла
}
```

```
for(тип переменная : последовательность)
{
   инструкции;
}
```

```
while(условие)
{
   // выполняемые действия
}
```

Цикл do гарантирует хотя бы однократное выполнение действий

```
do
{
   инструкции
}
while(условие);
```

Полезности от меня)

04

Курсы из разных областей

- **Фронтенд**

  https://rs.school/courses/javascript-preschool-ru

  https://github.com/rolling-scopes-school/tasks/tree/master/stage0

- ML

  1. **Нейронные сети и компьютерное зрение** Samsung

     https://stepik.org/course/50352/info

  2. **Нейронные сети и обработка текста** Samsung

     https://stepik.org/course/54098/info

  3. DLS Deep learning School

     https://stepik.org/org/dlschool

- **Бекэнд**

  Минимальный набор для участия в хакатонах:

  1. **Написание** API
  Python – FastApi
  https://fastapi.tiangolo.com/

  2. **Работа с хранилищами**
  Postgres, S3, MongoDb

  3. Docker
  https://karpov.courses/docker

Домашнее задание

Контест ДЗ1