Белорусский государственный университет информатики и радиоэлектроники

Основы конструирования программ

Преподаватель: **Меженная Марина Михайловна**

к.т.н., доцент,
доцент кафедры инженерной психологии и эргономики
а 609-2
mezhennaya@bsuir.by

Лекция 2: Качественный код. История развития и классификация языков программирования.

План лекции:

1.C++ Code Convention и правила создания качественного кода.

2.История возникновения и развития языков программирования.

3.Классификация языков программирования.

C++ Code Convention

```cpp
#include<iostream>
using namespace std;
int main(){ setlocale(LC_ALL, "rus");
int boxWithFruit=15;//количество ящиков на складе
int amountBoxForSale=0;// количество отгружаемых ящиков
cout<<"На складе сейчас"<<boxWithFruit<<" ящиков с фруктами.\n\n";
for(int i=1;;i++)//i - количество машин к погрузке
{cout<<"Сколько ящиков отгрузить в "<<i<<"-ю машину? ";
cin>>amountBoxForSale;
if(amountBoxForSale>boxWithFruit)
{cout<<"\nНа складе нет такого количества товара!";
cout<<"Осталось только "<<boxWithFruit<<" ящиков\n\n";
i--;//уменьшить счетчик на 1
}else{boxWithFruit-=amountBoxForSale;//перезаписываем значение
cout<<"Осталось "<<boxWithFruit<<" ящиков.\n";
}if(boxWithFruit==0)//если ящиков больше нет - выйти из цикла
{cout<<"Фрукты закончились! До свидания!\n";
break;}}return 0;}
```

Плохо

C++ Code Convention

```cpp
#include<iostream>
using namespace std;

int main()
{
    setlocale(LC_ALL, "rus");

    int boxWithFruit = 15; // количество ящиков на складе
    int amountBoxForSale = 0; // количество отгружаемых ящиков

    cout << "На складе сейчас" << boxWithFruit << " ящиков с фруктами.\n\n
    for (int i = 1;; i++) //  i - количество машин к погрузке
    {
        cout << "Сколько ящиков отгрузить в " << i << "-ю машину? ";
        cin >> amountBoxForSale;

        if (amountBoxForSale > boxWithFruit)
        {
            cout << "\nНа складе нет такого количества товара!";
            cout << "Осталось только " << boxWithFruit << " ящиков\n\n";
            i--; // уменьшить счетчик на 1
        }
        else
        {
            boxWithFruit -= amountBoxForSale; // перезаписываем значение
            cout << "Осталось " << boxWithFruit << " ящиков.\n";
        }

        if (boxWithFruit == 0)// если ящиков больше нет - выйти из цикла
        {
            cout << "Фрукты закончились! До свидания!\n";
            break;
        }
    }
    return 0;
```

C++ Code Convention:
Именование переменных, констант

Имя переменной – имя существительное, всегда начинается с буквы нижнего регистра, допускается нижнее подчеркивание и «верблюжья» нотация:

name
age
box_with_apple
boxWithApple

Имя константы – имя существительное, должно состоять только из букв верхнего регистра и нижнего подчеркивания:

HOURS_IN_DAY
SIZE

C++ Code Convention:
Именование методов (функций), классов, файлов

Имя класса – имя существительное, всегда начинается с буквы верхнего регистра, для составных названий используется «верблюжья» нотация (нельзя использовать нижнее подчеркивание):

class User

class MyAdapter

Имя метода (функции) – глагол+существительное, всегда начинается с буквы нижнего регистра, используется «верблюжья» нотация (нельзя использовать нижнее подчеркивание):

printData()

setName()

C++ Code Convention:
Именование методов (функций), классов, файлов

Имя файла должно состоять только из букв нижнего регистра и цифр, допускается нижнее подчеркивание:

io_base.cpp

user_data.h

C++ Code Convention
Именование файлов, переменных, констант, функций, классов

Главный принцип – дать переменной (функции, классу и т.д.) **осмысленное имя**, как можно более близкое к контексту использования:
age – возраст
number – номер
amount – количество
name – имя

Имена следует писать не английским транслитом, а английскими словами:
~~vozrast~~ → age
~~kolichestvo~~ → amount

C++ Code Convention
Фигурные скобки

Рекомендовано использовать фигурные скобки в блоках if, else, while, do, for даже если они содержат всего одну строку. Например:

```C++
1 for(int example = 0; example < 10; example ++)
2 {
3     cout << example << end;
4 }
```

Каждую фигурную скобку желательно располагать в отдельной строке. Так очень легко проследить, где блок начинается и где заканчивается.

C++ Code Convention
Пробелы в строке

При использовании оператора присвоения значения и операторов арифметических операций пробелы необходимы с обоих сторон от этого оператора:

```C++
variable = a + b - c;
variable=a+b-c; // слитно выглядит так
```

```C++
variable = -4;

variable++;
```

Исключение – унарные операторы.

C++ Code Convention
Табуляция

```cpp
int main()
{
for (int i = 0; i < 12; i++)
{
for (int j = 0; j < 12; j++)
{
cout << '@';
}
cout << endl;
}
return 0;
}
```

без табуляции

с табуляцией

```cpp
int main()
{
	for (int i = 0; i < 12; i++)
	{
		for (int j = 0; j < 12; j++)
		{
			cout << '@';
		}
		cout << endl;
	}
	return 0;
}
```

C++ Code Convention

В Microsoft Visual Studio есть “спасательная комбинация клавиш” **Ctrl+K затем Ctrl+F**, нажав которую осуществится **форматирование** выделенного исходного кода.

C++ Code Convention комментирование кода

Журналист спрашивает программиста:
— Какой код вы бы назвали плохим?
— Без комментариев.

Комментарии играют важную роль в поддержании читаемости кода!

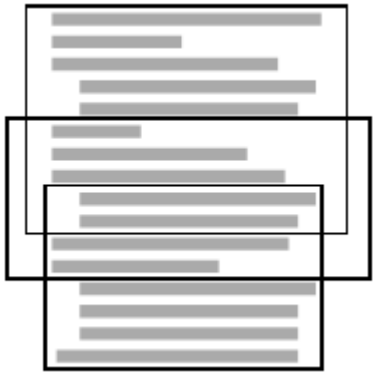
Оставлять комментарии в коде можно так:
// (комментирование одной строки),
/* комментарий */ (многострочный комментарий).

```
/*
int number;                                    // декларация переменной
cout << "Enter number: " << endl;     // вводим числа с клавиатуры
cin >> number;  // запись введенного числа в переменную number
*/
```

Принципы организации кода

Главный принцип организации последовательного кода – **группировать взаимосвязанные выражения** в блоки и отделять их в коде **пустой строкой**. Взаимосвязанные выражения работают с одними и теми же данными, выполняют схожие задачи или зависят от порядка выполнения друг друга.

Хорошо организованный код



Неудачно организованный код

Антипаттерн проектирования: магические числа

Код не должен содержать неименованных числовых констант (так называемых «магических» чисел), неименованных строковых констант (например, имен файлов и др.).

Подобного рода информацию следует выносить в глобальные переменные с атрибутом const.

По правилам хорошего стиля программирования тексты всех информационных сообщений, выводимых пользователю в ответ на его действия, также оформляются как константы.

Принципы организации условных операторов
Оператор if

Размещайте **наиболее вероятные** варианты раньше остальных. Так вы увеличите производительность программы, потому что уменьшите число проверок, выполняемых кодом в большинстве случаев.

Убедитесь, что при сравнении **на равенство** ветвление корректно. Использование > вместо >= или < вместо <= это риск ошибки!

Избегайте вложенности if **более трех.**

Сначала напишите код номинального хода алгоритма, затем опишите исключительные случаи.

Принципы организации условных операторов
Оператор switch

Организовать **порядок следования** вариантов можно по-разному:

- по алфавиту или численно, если все варианты равновероятны;
- правильный вариант → первый, далее – исключения;
- варианты по вероятности появления.

Сделайте обработку каждого варианта **простой**. Код, связанный с каждым вариантом, должен быть коротким. Если действия, предпринимаемые для какого-то варианта слишком сложны, напишите **функцию/метод** и вызывайте его.

Используйте вариант по умолчанию (**default**) только для обработки настоящих значений по умолчанию, но не кодируйте последний оставшийся вариант как вариант по умолчанию.

Принципы организации операторов цикла

Правила выбора вида цикла:

Если вы заранее **не знаете**, сколько итераций должен выполнить цикл, используйте **while.**

Если Вы точно знаете, что цикл должен выполняться **хотя бы раз**, то используйте вариант **do while.**

Цикл **for** – хороший вариант, если вам нужен цикл, выполняющийся **определенное количество раз**.

Принципы организации операторов цикла

Оптимальная длина цикла:

Делайте циклы достаточно короткими, чтобы их можно было увидеть сразу целиком. Эксперты предложили ограничивать длину цикла одной страницей. Однако когда вы оцените преимущество создания простого кода, вы редко будете писать циклы длиннее 15 или 20 строк.

Ограничивайте вложенность тремя уровнями. Если вам нужно большее число уровней, сделайте цикл короче (зрительно), вынося его часть в отдельный метод.

Принципы организации операторов цикла

Бесконечный цикл реализуйте следующим образом:

```
while (true)
{
   …
   if (условие)
      {
        break;
       }
   …
}
```

Принципы организации операторов цикла: работа с переменными-счетчиками

Используйте смысловые имена переменных-счетчиков, чтобы сделать вложенные циклы читабельными.

```
for ( int i = 0; i < numPayCodes; i++ ) {
   for ( int j = 0; j < 12; j++ ) {
      for ( int k = 0; k < numDivisions; k++ ) {
         sum = sum + transaction[ j ][ i ][ k ];
      }
   }
}
```

Плохо

```
for ( int payCodeIdx = 0; payCodeIdx < numPayCodes; payCodeIdx++ ) {
   for (int month = 0; month < 12; month++ ) {
      for ( int divisionIdx = 0; divisionIdx < numDivisions; divisionIdx++ ) {
         sum = sum + transaction[ month ][ payCodeIdx ][ divisionIdx ];
      }
   }
}
```

Хорошо

Принципы организации операторов цикла: работа с переменными-счетчиками

Цикл while:
Располагайте служебные операции либо в начале, либо в конце цикла. Служебные операции цикла — это выражения вроде i = i + 1 или j++, чье основное назначение не выполнять работу в цикле, а управлять циклом.

Цикл for:
Когда вы используете цикл for, манипуляции с его счетчиком в теле цикла должны быть под запретом. Для таких случаев используйте цикл while.

Правила работы с массивами в C++

- индексация с 0 до n-1;
- имя массива является адресом первого байта его первого элемента;
- не производится проверка выхода за границы массива (любая попытка проверки делает язык более медленным);
- для динамически создаваемых массивов: необходимо освобождать выделенную память (в противном случае – утечки памяти).

Принципы использования функций

Код не должен дублироваться – для этого существуют функции!

Более того: все логически завершенные фрагменты кода должны быть оформлены в виде функций!

Одна функция решает только одну задачу (например, не допускается в одной функции считывать данные из файла и выводить их на консоль – это две разные функции!). При этом внутри функции возможен вызов других функций.

Если код Вашей функции более 50 строчек – скорее всего, Вы не до конца разбили ее на более мелкие функции.

Типичные ошибки начинающих

Типичные ошибки начинающих

- Не знание синтаксиса базовых конструкций языка: if-else, switch-case, while, do-while, for, правил работы с функциями.
- Пропуск ; в конце инструкции или другая крайность – размещение ; там, где это не нужно.
- Путаница в количестве открывающихся{ и закрывающихся } скобок (пишите обе скобки одновременно!).
- Использование в условии знака присвоения (=) вместо ==.
- Ни о чем не говорящие имена переменных и функций.

Типичные ошибки начинающих

Задачка:

Студенту-первокурснику (голодному) дали три яблока. Он съел два яблока. Сколько яблок осталось у студента-первокурсника?

1. Одно
2. У него не осталось яблок – он же был голодный…
3. Никто не знает наверняка – что за глупые вопросы на лекции по ОКП?

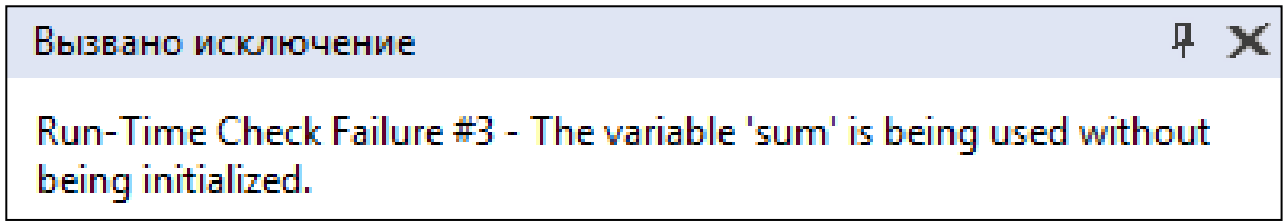
Типичные ошибки начинающих

Задачка:

Студенту-первокурснику (голодному) дали три яблока. Он съел два яблока. Сколько яблок осталось у студента-первокурсника?

- А сколько яблок у него было изначально? Неизвестно…

Поэтому нужно обнулять переменные!!!



Типичные ошибки начинающих

- Объявление служебных переменных и попытка их использования без инициализации (без присвоения начального значения).
- «Магические» числа и строки в коде.
- Отсутствие освобождения памяти (с помощью оператора delete) после ее ручного выделения (с помощью оператора new).
- Использование слишком длинных функций, которые в реальности следовало бы разбить на несколько функций.
- Сложная (неочевидная, неоптимальная) логика решения задачи.
- Аргументация «Но программа же работает!»

Типичные ошибки начинающих: итог

Программируем, как умеем!

~~**Программирование – это магия**~~

Языки программирования: исторический очерк

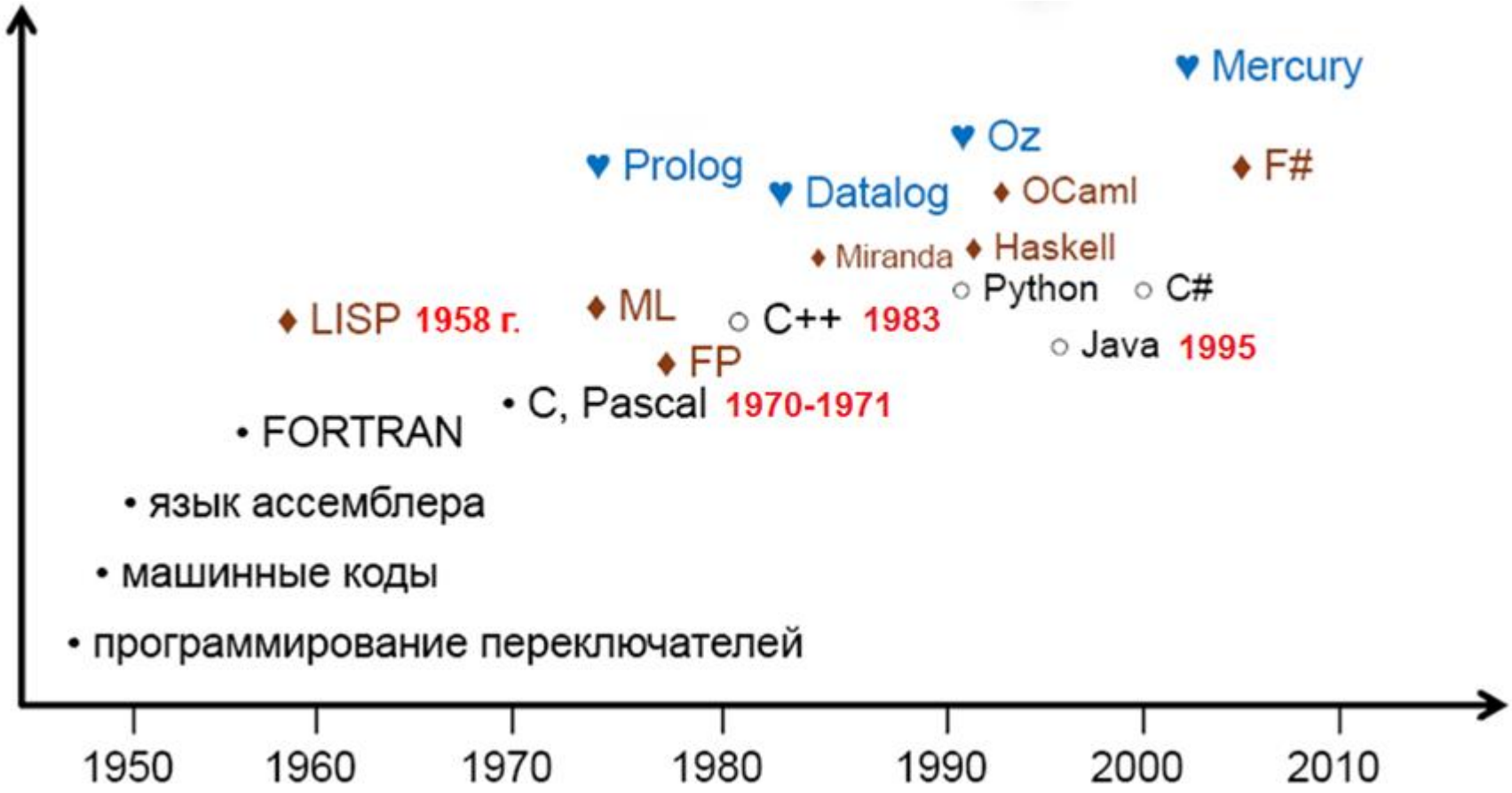

```
0000  0A 12 1F 4B C3 E0 EE F1
0008  C3 1D 23 17 F2 00 0C 0D
0010  …
```

```
            MOV       AX, [ARG1]
            ADD       AX, [ARG2]
            MOV       [RES], AX
            JMP       NEXT
ARG1:       DB        10
ARG2:       DB        20
RES:        DB        0
NEXT:       …
```

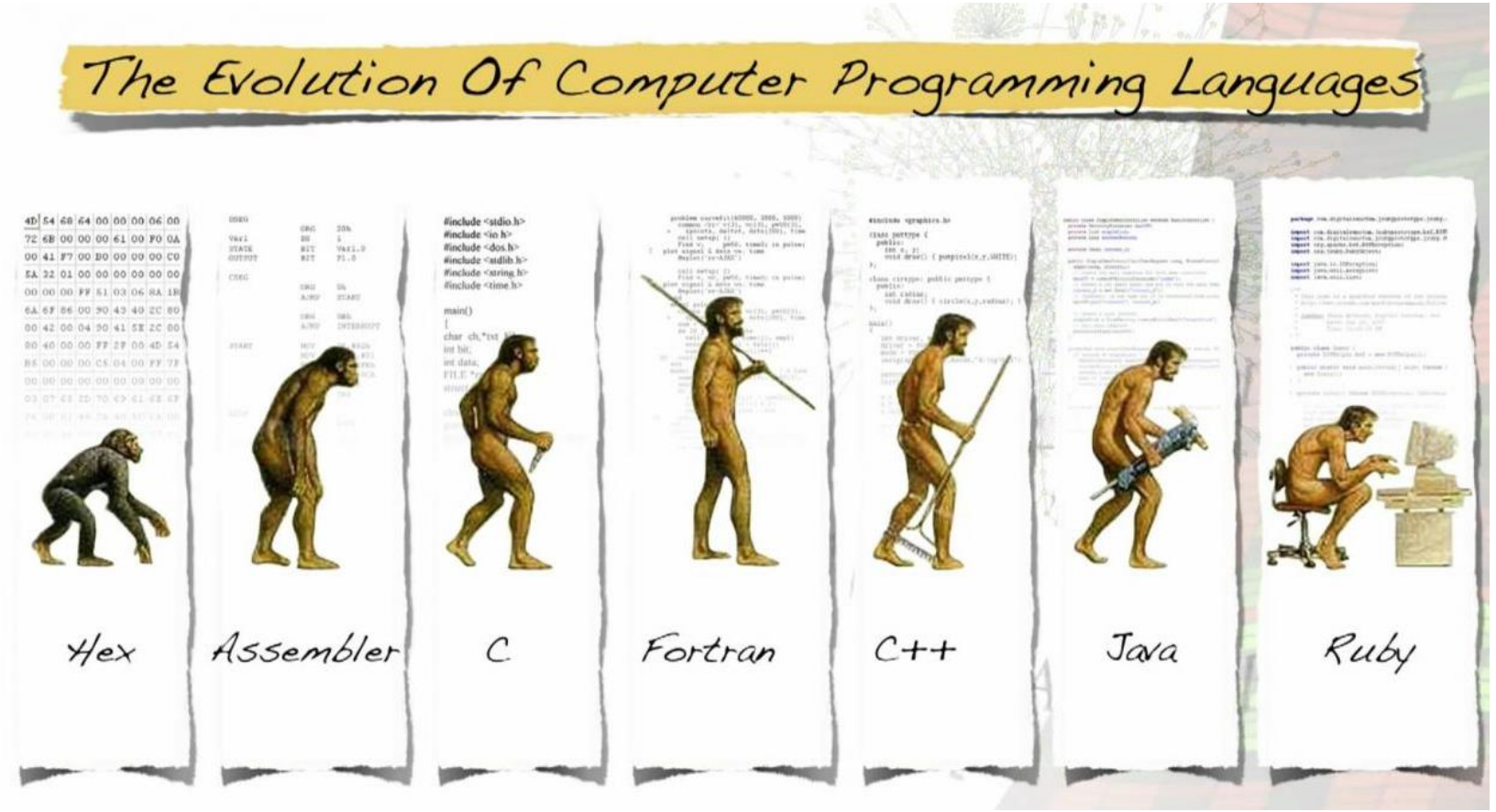
```
            S = 0
            DO 10 I=1,10
            S = S + I*I
10          CONTINUE
```

- Первый язык программирования высокого уровня – ФОРТРАН – был создан Дж.Бэкусом, чтобы математики могли программировать на уровне формул.

Языки программирования: исторический очерк



Языки программирования: исторический очерк



Стили программирования: императивный и декларативный.

Парадигмы программирования: процедурная (структурная), объектно-ориентированная, функциональная, логическая.

Стили и парадигмы программирования

Парадигма программирования - исходная концептуальная схема постановки проблем и их решения.

Вместе с языком, ее формализирующим, парадигма формирует **стиль программирования.**

Парадигма отражает то, как программист видит выполнение программы.

Например, в объектно-ориентированном программировании программист рассматривает программу как набор взаимодействующих объектов, тогда как в функциональном программировании программа представляется в виде цепочки вычисления функций.

Стили программирования: императивный

Программируя в императивном стиле, программист должен объяснить компьютеру, **как нужно решать задачу.**

Императивная программа это последовательность команд / логических переходов, которые должен выполнить компьютер.

Основные конструкции:

- Манипулирование ячейками памяти
- Оператор присваивания
- Явные операторы передачи управления
- Условный оператор, циклы

Стили программирования: декларативный

Программа представляет собой совокупность утверждений, описывающих фрагмент предметной области или сложившуюся ситуацию; **описывается результат (его свойства), а не методы его достижения.**

Программист лишь описывает решаемую задачу, а поиском решения занимается императивная система программирования.
В итоге получаем значительно большую скорость разработки приложений, значительно меньший размер исходного кода.

Парадигмы программирования: процедурная

Программа состоит из структур данных (**объектов обработки**) и алгоритма (**метода обработки**).

Программист должен **в явном виде** описать все вычисления, которые должен проделать компьютер.

Для управления процессом выполнения используются следующие конструкции: последовательность, ветвление, цикл и вызов подпрограммы.

Эта парадигма является **самой старой.**

Парадигмы программирования: функциональная

Функциональные языки **оперируют данными.** Применение функции к аргументам изменяет данные.

Единственной управляющей конструкцией является **вызов функции.** Используется **рекурсия.**

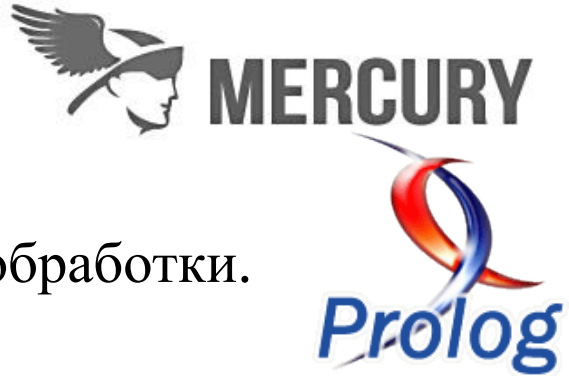
Сформировались для развития систем искусственного интеллекта.

Парадигмы программирования: логическая

Логическая программа оперирует **пространством поиска решений.**

Программа представляет собой **набор отношений**, которые называются фактами, и **правил**, на основании которых могут быть получены новые отношения. Это своего рода база данных (БД). Ее применение инициализируется запросом. Поиск ответа заключается в попытке логического вывода на основании фактов и правил, имеющихся в БД.

Возникло как упрощение функционального программирования для математиков и лингвистов, решающих задачи символьной обработки.



Парадигмы программирования: объектно-ориентированная

В объектно-ориентированном подходе исходная задача представляется как **совокупность взаимодействующих объектов**.

По аналогии с реальным миром объект характеризуется совокупностью **свойств** (структур данных), **методов их обработки** и **событий**, на которые данный объект может реагировать и которые приводят, как правило, к изменению свойств объекта.

Объекты могут иметь идентичную структуру и отличаться только значениями свойств. В таких случаях в программе создается новый тип данных – **класс.**

Базовые понятия ООП

Класс — это тип данных, вводимый пользователем.
Основное назначение класса - описание основных свойств и методов обработки объектов этого типа данных.
С точки зрения программирования класс можно рассматривать как набор данных (полей, атрибутов, членов класса) и функций для работы с ними (методов).

Объект (экземпляр класса) – это отдельный представитель класса, имеющий конкретное состояние и поведение, полностью определяемое классом.

То есть класс – это логическая конструкция, а объект обладает физической сущностью (т.е. объект занимает область в памяти).

Базовые понятия ООП

Базовые понятия ООП

1.Абстракция данных – выделение только необходимых (существенных для поставленной задачи) характеристик и методов, которые в полной мере описывают объект.

2.Инкапсуляция – объединение данных и методов (функций) для работы с ними, а также сокрытие внутренней реализации класса от пользователя.

3.Наследование – механизм, позволяющий описать новый класс на основе уже существующего (родительского), при этом свойства и функциональность родительского класса частично или полностью заимствуются новым классом.

4.Полиморфизм – возможность использования одних и тех же методов для работы с различными объектами базового и порожденного им классов.

Достоинства ООП

- Абстракция от деталей реализации.
- Данные и операции описываются вместе.
- Модульность (локализация кода и данных улучшает наглядность).
- Управление сложностью программы за счет сокрытия данных, методов.
- Возможность создавать расширяемые системы.
- Возможность многоразового использования программных компонентов.

Недостатки ООП

- Сложность понимания концепций ООП.
- Сложность проектирования и использования классов.
- Излишняя универсальность.
- Неэффективность на этапе выполнения (инкапсуляция данных приводит к необходимости выполнения процедурного вызова каждый раз при доступе к данным).

Процедурное программирование или ООП?

1.Является ли **проблема** статичной или динамичной (будут ли изменения и новые версии)? ООП для статической проблемы будет мешать, т.к. не покажет всю картину целиком.

2.Является ли Ваше **решение** одно- или многоуровневым? ООП хорош, если необходимо разбивать проблему сверху вниз и нужно задумываться о каждом конкретном шаге.

3.Глобальное или локальное **состояние** программы? Состояние определяется статусом переменных. Если везде статические переменные, то зачем нужен ООП?

Классификация языков программирования по уровню абстракции

1 уровень: машинный
(язык машинных кодов)

2 уровень: машинно-ориентированный
(Assembler, C, C++)

машинно-зависимые
(низкого уровня)

3 уровень: структурные
(процедурно-ориентированные и объектно-ориентированные)
(C++, C#, Java, …)

4 уровень: проблемно-ориентированные
(Fortran, Lisp)

5 уровень: ???

машинно-независимые
(высокого уровня)

Классификация языков программирования по структуре кода

1.Неструктурные (используют операторы goto или jump: – языки машинного и машинно-ориентированного уровней)

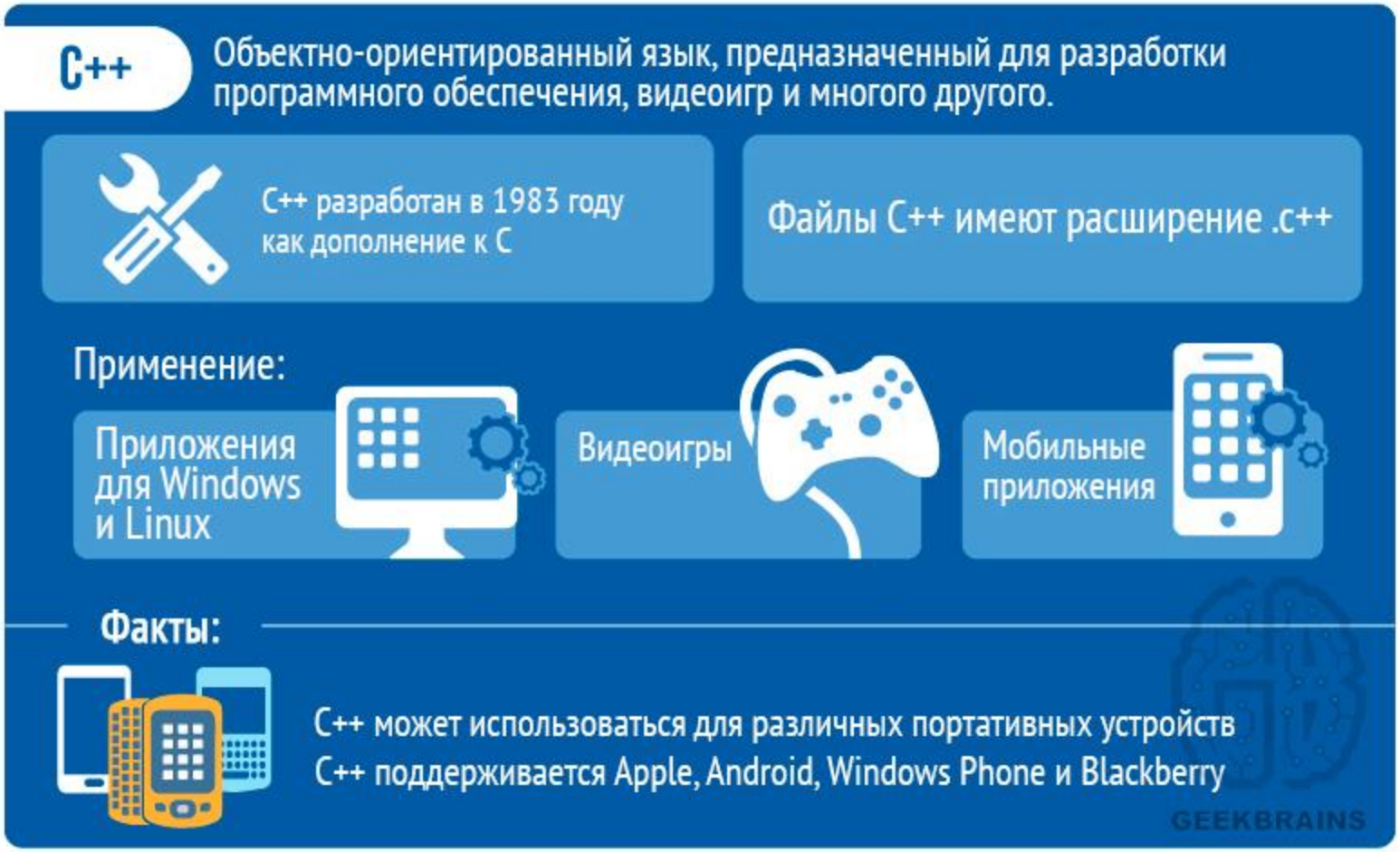
2.Структурные

3.Объектно-ориентированные

4.Визуальные

Что дальше?


С чего начать изучение программирования?
ЧТО ТЫ ХОЧЕШЬ ДЕЛАТЬ?
Самый простой для изучения: Python
Самый мощный: C++
Наверняка останется востребованным и через 10 лет: Java
Твоя конечная цель определит набор языков для изучения:
СОЗДАНИЕ И ОФОРМЛЕНИЕ СТАТИЧНЫХ СТРАНИЦ
HTML + CSS
СОЗДАНИЕ ИНТЕРАКТИВНЫХ ВЕБ-СТРАНИЦ
JavaScript
РАБОТА С ИНФОРМАЦИЕЙ (СЕРВЕРНАЯ РАЗРАБОТКА)
PHP
Python
Ruby
РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ
Swift,Obj C (iOS)
Java, C++ (Android)


C++
Объектно-ориентированный язык, предназначенный для разработки программного обеспечения, видеоигр и многого другого.
C++ разработан в 1983 году как дополнение к C
Файлы C++ имеют расширение .c++
Применение:
Приложения для Windows и Linux
Видеоигры
Мобильные приложения
Факты:
C++ может использоваться для различных портативных устройств
C++ поддерживается Apple, Android, Windows Phone и Blackberry
GEEKBRAINS

JAVA®

Это серверный, интерпретируемый, компилируемый язык, выполняющийся на виртуальных машинах. Не имеет отношения к JavaScript.

Был разработан в 1995 году; один из старейших языков программирования для web.

Применение:

Онлайн игры

Загрузка фото

Виртуальные туры

Интерактивные карты

Факты:

Пользователи могут отключить Java на своих машинах

Является основой Android

GEEKBRAINS

PHP

Это интерпретируемый, некомпилируемый, скриптовый серверный язык.
Поскольку команды выполняются на сервере, отображается результат в простом HTML.

- Создан в 1994 году Расмусом Лерфордом
- Впервые представлен публике: 1995
- 2.0: 1996
- 3.0: 1997/1998
- 4.0: 1999/2000
- 5.0: Июль 2004

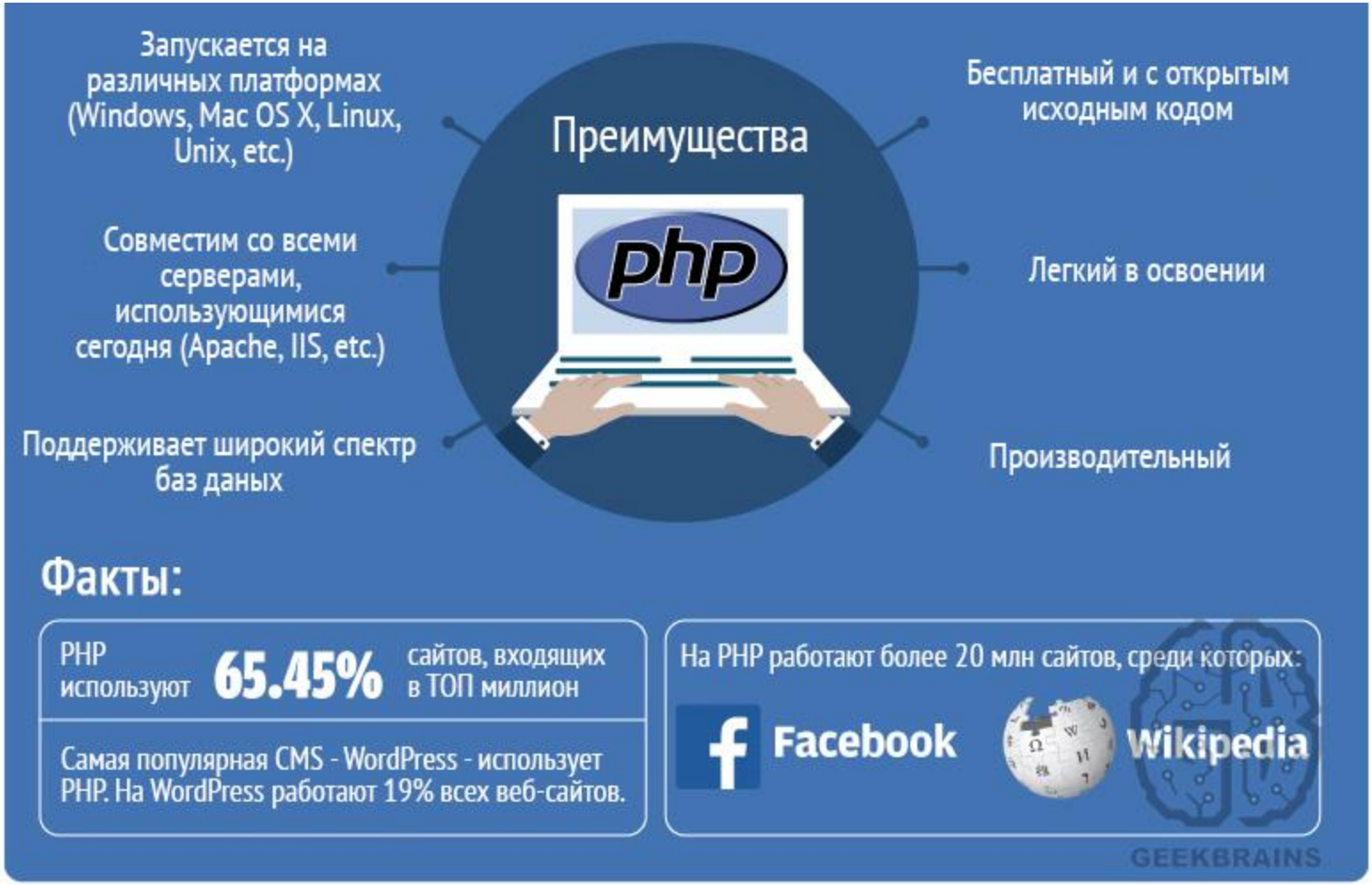
Файлы имеют расширение .php

Поддерживает:

- Text
- HTML
- CSS
- JavaScript
- PHP code

Возможности:

- Создание динамических страниц
- Работа с файлами на сервере
- Работа с формами
- Отправление и получение кук
- Изменение данных в БД
- Ограничение действий пользователя на странице
- Шифрование даных


Преимущества
Запускается на различных платформах (Windows, Mac OS X, Linux, Unix, etc.)
Совместим со всеми серверами, использующимися сегодня (Apache, IIS, etc.)
Поддерживает широкий спектр баз даных
Бесплатный и с открытым исходным кодом
Легкий в освоении
Производительный
php
Факты:
PHP используют 65.45% сайтов, входящих в ТОП миллион
Самая популярная CMS - WordPress - использует PHP. На WordPress работают 19% всех веб-сайтов.
На PHP работают более 20 млн сайтов, среди которых:
Facebook
Wikipedia
GEEKBRAINS

JAVASCRIPT

Клиентский скриптовый язык. Поддерживается практически всеми браузерами.

Разработан в 1995 году компанией Netscape®

Использование:


Реклама


Аналитика


Виджеты

JQuery – самая популярная библиотека JS, применяется на

22.9%

сайтов из ТОП-миллион

Она позволяет:

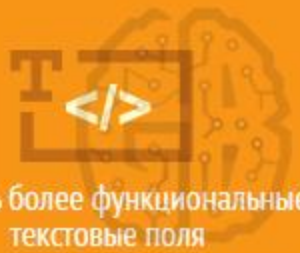
- Игнорировать различия JS браузеров
- Облегличть процесс разработки

С помощью JS вы можете:


проверить существование имени пользователя при регистрации без перезагрузки страницы


сделать функцию автодополнения ввода на вашем сайте


поправить ошибки верстки



сделать более функциональные текстовые поля

GEEKBRAINS

ASYNCHRONOUS JAVASCRIPT AND XML (AJAX)

Это технология, позволяющая использовать несколько языков вместе (не являясь языком)..

AJAX позволяет коду на JS общаться с сервером в фоне и менять содержимое страницы без ее обновления (f5)

На сегодняшний день интернет трудно представить без AJAX. Он используется:

Во всех браузерных чатах (как Google Talk)

В интерфейсе Gmail

В Facebook для чата и работы с фото

Вы не сможете использовать AJAX должным образом, не зная JavaScript и CSS

Факты:

STRUCTURED QUERY LANGUAGE (SQL)

SQL — это не язык программирования или разметки. Он является стандартом для взаимодействия с Базами Данных: сохранения и обработки информации. Самая распространенная БД (и диалект SQL) — MySQL, рекомендуется начинающим разработчикам.

Был разработан в **1979**

Файлы SQL имеют расширение **.sql**

Возможности

- Запросы к Базе даных
- Получение данных
- Вставка записей
- Обновление записей
- Удаление записей
- Создание новых баз
- Создание новых таблиц
- Создание хранимых процедур
- Создание представления в БД
- Управение правами доступа

Факты:

С SQL работают такие компании как

ORACLE, SYBASE, Microsoft

Их решения довольно сильно различаются по типам задач, но все поддерживают стандарт языка SQL. Самая популярная версия — MySQL (она бесплатна и с открытым исходным кодом).

OBJECTIVE C

Объектно-ориентированный язык; используется для разработки под продукты Apple

Objective C был разработан в **1990** году

Файлы Objective C имеют расширение .m

Применение:

Приложения для iOS

iOS

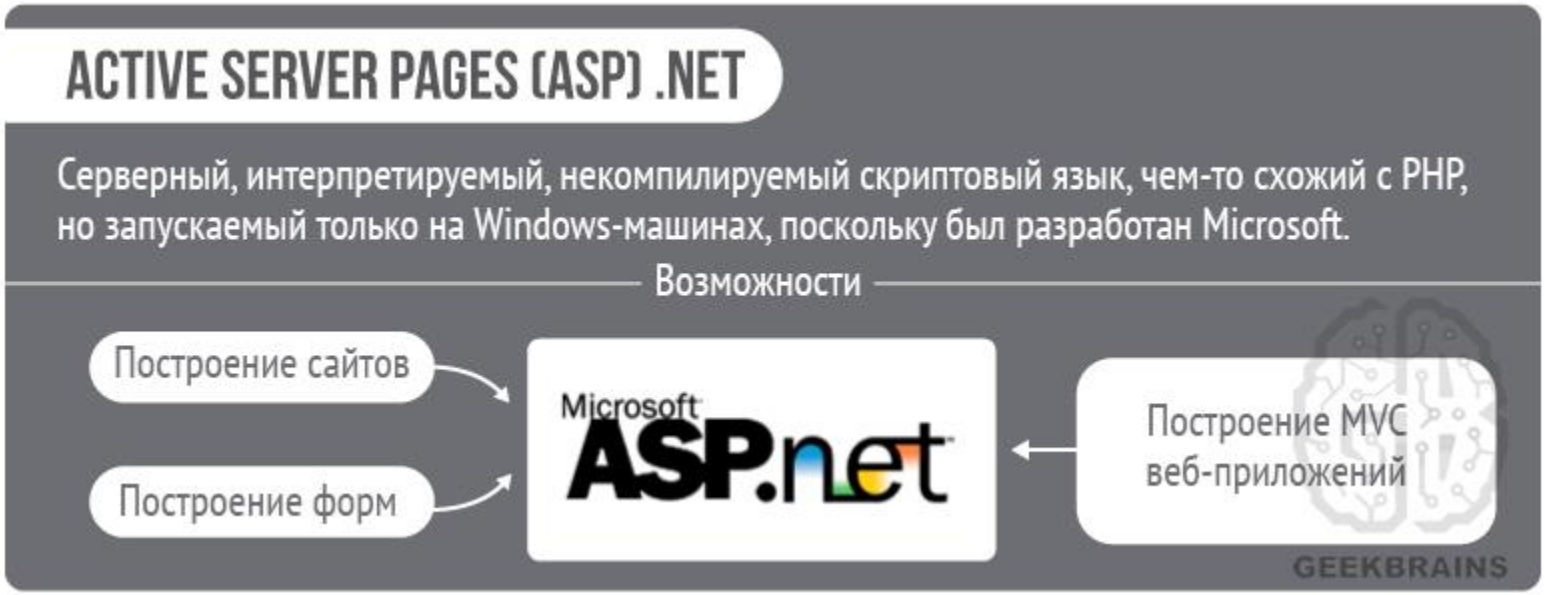
Программное обеспечение для Mac OS

Факты:

Objective C предназначен только для разработки Apple-приложений, и не подходит для Android.

GEEKBRAINS


PYTHON™
Серверный, интерпретируемый, некомпилируемый скриптовый язык с открытым исходным кодом. Может использоваться самостоятельно или в составе другого фреймворка, например, django.
Создание сайтов
Построение графических пользовательских интерфейсов
python™
Обеспечение доступа к БД
ПО и игры
Факты:
Используется подрядчиком NASA (United Space Alliance)
Легок в освоении по сравнению с C++
GEEKBRAINS


ACTIVE SERVER PAGES (ASP) .NET
Серверный, интерпретируемый, некомпилируемый скриптовый язык, чем-то схожий с PHP, но запускаемый только на Windows-машинах, поскольку был разработан Microsoft.
Возможности
Построение сайтов
Построение форм
Microsoft ASP.net
Построение MVC веб-приложений
GEEKBRAINS

RUBY

Серверный, интерпретируемый, некомпилируемый скриптовый язык японского происхождения. Увидел свет в 1995 году.

Набирает популярность за счет того, что используется с Ruby on Rails, удобным фреймворком, подобным django или Python

Представляет собой сочетание языков:

Perl
Smalltalk
Eiffel
Ada
Lisp

Применение

Построение симуляторов

Создание веб-приложений

Факты:

Руби работает на многих платформах: UNIX'ы, Mac OS X, Windows 95/98/Me/NT/2000/XP, DOS, BeOS, OS/2...
У руби есть много реализаций:
JRuby – интерпретатор написанный целиком на Java
Rubinius – интерпретатор Ruby на Ruby (!)
MacRuby – тесно интегрирован с Cocoa, используется в программах для Mac OS