FIAP

Sintaxe Objective-C

X-Code com objective-C

Prof. Agesandro Scarpioni
agesandro@fiap.com.br

Tipos de dados primitivos

- int, float, double, char e bool
- Para criarmos uma variável inteira e armazenarmos a idade de uma pessoa utilizamos:
  - int idade;
- Para criarmos uma variável com casas decimais e armazenarmos a altura de uma pessoa utilizamos:
  - float altura;

Tipos de dados primitivos

- Podemos declarar uma variável e em outra linha atribuir um valor, ou na mesma linha declarar a variável e atribuir um valor.
    - int altura; // declarando
    - altura = 1.87; // atribuindo
    - int altura = 1.87; // declarando e atribuindo

Tipos de dados inteiros

- Por padrão uma variável inteira pode armazenar números inteiros negativos e positivos, porém se quisermos restringir o uso de números negativos utilizamos o qualificador unsigned na criação:
  - unsigned int idade;
- Por padrão toda variável inteira é criada com signed, por este motivo não é comum a declaração deste qualificador, ex: (signed int idade).
- O int é um tipo padrão para as variáveis e parâmetros inteiros, portanto, você poderia declarar a idade da seguinte forma:
  - unsigned idade;

Tipos de dados inteiros

- As constantes inteiras por padrão são especificadas em base 10, ou seja, em números decimais, porém, também é possível especificar tais constantes com números de outras bases utilizando um prefixo à frente. Todas as constantes no exemplo abaixo estão armazenando o número decimal 15.
  - int IDADE = 15; // decimal base 10 e não utilizamos um prefixo
  - int IDADE = 017; // octal base 8 e usamos o prefixo 0 (zero)
  - int IDADE = 0x0F; // Hexadecimal base 16, o prefixo é 0x (zero x)

FIAP

Tipos de dados inteiros

- O fato de usarmos unsigned pode ser útil para dobrarmos a capacidade de armazenamento da variável, por exemplo uma variável inteira que poderíamos armazenar com sinal o intervalo de números entre -32768 até 32767, sem sinal poderíamos armazenar o intervalo entre 0 e 65535, além dos qualificadores unsigned e signed temos os qualificadores short e long.

Tipo de Dado	Bits (tamanho)	Sem sinal	Com sinal
short int	16	0 a 65535	-32768 a 32767
int	32	0 a 4.294.967.295	-2.147.483.648 a 2.147.483.647
long int	32	0 a 4.294.967.295	-2.147.483.648 a 2.147.483.647
long long int	64	0 a (2 elev 64 -1)	-2 elevado a 63 até 2(elev 63 -1)

- obs: Falar sobre o tipo NSInteger e NSUInteger de 64 bits da biblioteca Foundation que pode funcionar como int ao invés de long int em plataformas de 32 bits e como inteiros de 64 bits em plataformas de 64 (ver slide 16). O long int suporta um espaço de endereçamento de 32 bits para plataformas de 32 bits, porém, para plataformas de 64 bits as variáveis long int e os endereços de memória são elevados para o tamanho de 64 bits, os outros tipos int permanecem o mesmo.

Ponto Flutuante

- Para declararmos variáveis de ponto flutuante temos dois tipos o float e double, lembrando que ponto flutuante significa que o ponto decimal pode ser colocado entre qualquer um dos dígitos significativos que formam o número.

- Constantes de ponto flutuante também podem ser expressas em notação científica ou exponencial, tanto x quanto y estão com o mesmo valor armazenado no exemplo abaixo:

    - float x = 0.0001;

    - float y = 1e-4; —> 1 x 10 potência de -4

Tipo de Dado	Bits (tamanho)	Sem sinal	Digitos significativos
float	32	+-1,5 x 10 elev-45 a +-3,4 x 10 elev 38	7
double	64	+-5,0 x 10elev-324 a +-1,7 x 10 elev 308	15

Caracteres

- Um tipo char pode armazenar um único caracter como a letra A, o número 3 ou um asterisco.
- Constantes de caracter podem ser atribuídas como um inteiro de 8 bits (veja a tabela ASCII) ou um caracter entre apóstrofes (não aspas), nas duas formas estamos armazenando o caracter A em maiúsculo.
    - char x = 'A';
    - char y = 65; // este é o número ASCII da letra A
- Por default char é um valor não-sinalizado e pode armazenar valores entre 0 e 255, quando utilizamos o qualificador signed podemos armazenar um valor entre -128 a 127, se quisermos armazenar números é melhor utilizarmos int.

String

- No Objective-C existem dois tipos de declaração de String, uma herdada da linguagem C e um novo tipo de String orientada a objetos da biblioteca Foundation chamada NSString.
- Constantes de string são representadas por aspas e não apóstrofes, para declarar uma constante de string usamos *.
    - char *frase = "Bom dia";
- Para uma constante de string utilizando NSString da biblioteca foundation, usamos @ antes do texto além do * que devemos utilizar sempre que utilizamos um objeto, esta característica define um PONTEIRO.
    - NSString *frase=@"Bom dia";

DICA: Para String prefira o tipo NSString ao invés de char é muito mais prático

Booleanos

- No Objective-C existe o BOOL que utiliza os valores YES e NO (TRUE e FALSE também podem ser utilizados), o BOOL não restringe apenas o armazenamento de 2 valores YES e NO o BOOL também é usado para outro tipo de dado, signed char.
    - BOOL ativo=YES;
- Quando o Objetive C avalia uma expressão booleana, ele prevê que qualquer valor "diferente de zero" indica verdadeiro e "zero" indica falso;

Booleanos

- Lembre-se: Quando o Objetive C avalia uma expressão booleana, ele prevê que qualquer valor "diferente de zero" indica verdadeiro e "zero" indica falso;

Veja o código abaixo:

```
BOOL valor=10; //Se vc colocar YES quando criar o BOOL irá passar no 1º
  IF,  se vc colocar NO ele passa no 2º IF, como colocamos o char 10 ele
  passo nos dois, melhor usar o !=YES

if (valor) {
NSLog(@"O resultado é verdadeiro"); // O NSLog equivale ao printf do C
}

if (valor != YES) {
NSLog(@"O resultado é falso"); // O NSLog equivale ao printf do C
}
```

DICA: Compare BOOL com NO ou seja !="YES" pois apenas um valor pode indicar o falso

Enumeradores

- Para definir um tipo de dado enumerado usamos enum, depois de enum damos um nome para lista de valores possíveis e os valores seguem dentro de uma chave, veja o exemplo abaixo:
  - enum sensacaoTermica { gelado, frio, morno, quente };
  - enum sensacaoTermica { gelado=1, frio, morno, quente};
  - enum temperatura { frio=9, quente=38 };

```
13
14      enum temperaturaCorporal {sem=36, febril=37, febre=38};
15
16      enum temperaturaCorporal valor;
17
18      valor = febril;
19
20      NSLog(@"A Temperatura é %u",valor);
21
```

```
2016-03-21 12:11:48.063 enum[1346:76642] A Temperatura é 37
Program ended with exit code: 0
```

DICA: O primeiro item de um enumerador tem valor 0, você pode alterar a sequência atribuindo um valor ao item, ou definir um valor diferente para cada item.

Enumeradores

- Outro exemplo:

```
enum febre {sem= 36, febril , febre, febrao, burnout};
enum febre temperatura;

NSLog(@"Entre com a sua emperatura 36,37,38,39,40 ?" );
scanf("%u", &temperatura);
switch (temperatura) {
    case 36:
        NSLog(@"Sem febre");
        break;
    case 37:
        NSLog(@"Febril");
        break;
    case 38:
        NSLog(@"Febre");
        break;
    case 39:
        NSLog(@"Febrão");
        break;
    case 40:
        NSLog(@"Burn Out");
        break;
    default:
        NSLog(@"Valor inválido");
        break;
}
```

NSLog

- O NSLog mostra uma String no console de depuração também pertence a biblioteca Foundation, ele equivale a um printf do C, uma String pode ter um ou mais marcadores de posição "placeholders", que são representados pelo caracter %, ou seja, para cada marcador o NSLog espera receber um argumento, desta forma ele substitui o(s) marcador(s) pelo(s) argumento(s). Após cada símbolo de "%" especificamos o tipo de dado do argumento, por exemplo: %d para inteiros sinalizados, %u para inteiros não sinalizados, %s ou %S para unichar.

Veja o código abaixo:

```
int idade = 30;
int trabalho = 15;

NSLog(@"Eu tenho %d anos e trabalho a %d anos", idade,trabalho);
```

NSLog

FIAP

Especificadores de formatos dos placeholders

Tipo de dados	Especificadores de formato
char	%c ou %C para unichar
char * (string c)	%s ou %S para unichar *
int signed	%d (decimal), %o (octal), %x (hexadecimal)
int unsigned	%u (decimal), %o (octal), %x (hexadecimal)
float / double	%e (científica), %f (decimal), %g
objeto	%@

%g neste especificador o NSLog determina o formato mais adequado conforme o valor a ser representado.

Existe um tipo id que pode armazenar uma referência a um objeto de qualquer tipo, veremos depois.

Use h para short, 1 (um) para long e 11 para long long. Ex. Variável long long int em decimal use %11d.
Variável short em decimal sem sinal use %hu.

Dica: Utilizando %% você insere o símbolo % na string, veja o exemplo abaixo:

```
float unsigned juros = 3.5;
NSLog(@"Atualmente o juros para empréstimo pessoal é de : %f%%", juros);
```

NSLog

Mais sobre especificadores de formatos

- Se você está trabalhando com NSInteger ou NSUInteger, onde o aplicativo desenvolvido vai rodar em aparelhos de 32 bits como o iPhone 5 ou anterior ou vai rodar em desktop de 64 bits, devemos ter alguns cuidados ao utilizar funções como o NSLog que aceitam tais especificadores. Veja o exemplo abaixo:
    - NSUInteger idade=60;
    - NSLog(@"A idade informada foi %u anos", idade);
- No Iphone a idade é um tipo int de 32 bits, no desktop de 64 bits a idade é um long de 64 bits, assim o especificador correto para os tipos NSInteger e NSUInteger de 64 bits sem sinal seria %lu e no iPhone %u, qual deles devemos usar ?

DICA: Faça a conversão do tipo para long assim sempre teremos um argumento de 64 bits independente da plataforma, assim o especificador %lu será o correto para ambas.
Exemplo: NSLog(@"A idade informada foi %lu anos", (long)idade);

NSLog

Alinhamento e formatação

- Para uma casa decimal com o argumento colado ao texto.

  float nota = 7.598;

  NSLog(@"A nota alcançada foi: %0.1f", nota);

  - Será apresentado: A nota alcançada foi: 7.6

OBS: O número após o percentual controla a quantidade de caracteres de alinhamento à direita, o número após o ponto controla a quantidade de casas decimais.

NSLog

Alinhamento e formatação

- Para 2 casas decimais com o argumento a 10 caracteres do texto, alinhado pela direita

```
float nota = 7.598;

NSLog(@"A nota alcançada foi: %10.2f", nota);
```

  - Será apresentado: A nota alcançada foi: 7.60

OBS: O número após o percentual controla a quantidade de caracteres de alinhamento à direita, o número após o ponto controla a quantidade de casas decimais.

NSLog

Alinhamento e formatação

- Para 2 casas decimais com o argumento a 10 caracteres do texto, alinhado pela direita com zeros à esquerda

  float nota = 7.598;

  NSLog(@"A nota alcançada foi: %010.2f", nota);

  - Será apresentado: A nota alcançada foi: 0000007.60

OBS: O número zero após o percentual informa que será preenchido os espaços em branco à esquerda com zeros

FIAP

NSLog

Alinhamento e formatação

- Para 2 casas decimais com o argumento a 10 caracteres do texto, alinhado pela esquerda

  float nota = 7.598;

  NSLog(@"A nota alcançada foi: %-010.2f parabéns", nota);

  - Será apresentado: A nota alcançada foi: 7.60 parabéns

OBS: O sinal de negativo após o percentual posiciona o alinhamento à esquerda

NSLog

Alinhamento e formatação

- Para 2 casas decimais com o argumento a 10 caracteres do texto, alinhado pela direita com zeros à esquerda e quebrando a linha

```
float nota = 7.598;

NSLog(@"A nota alcançada \n foi: %010.2f", nota);
```

  - Será apresentado: A nota alcançada

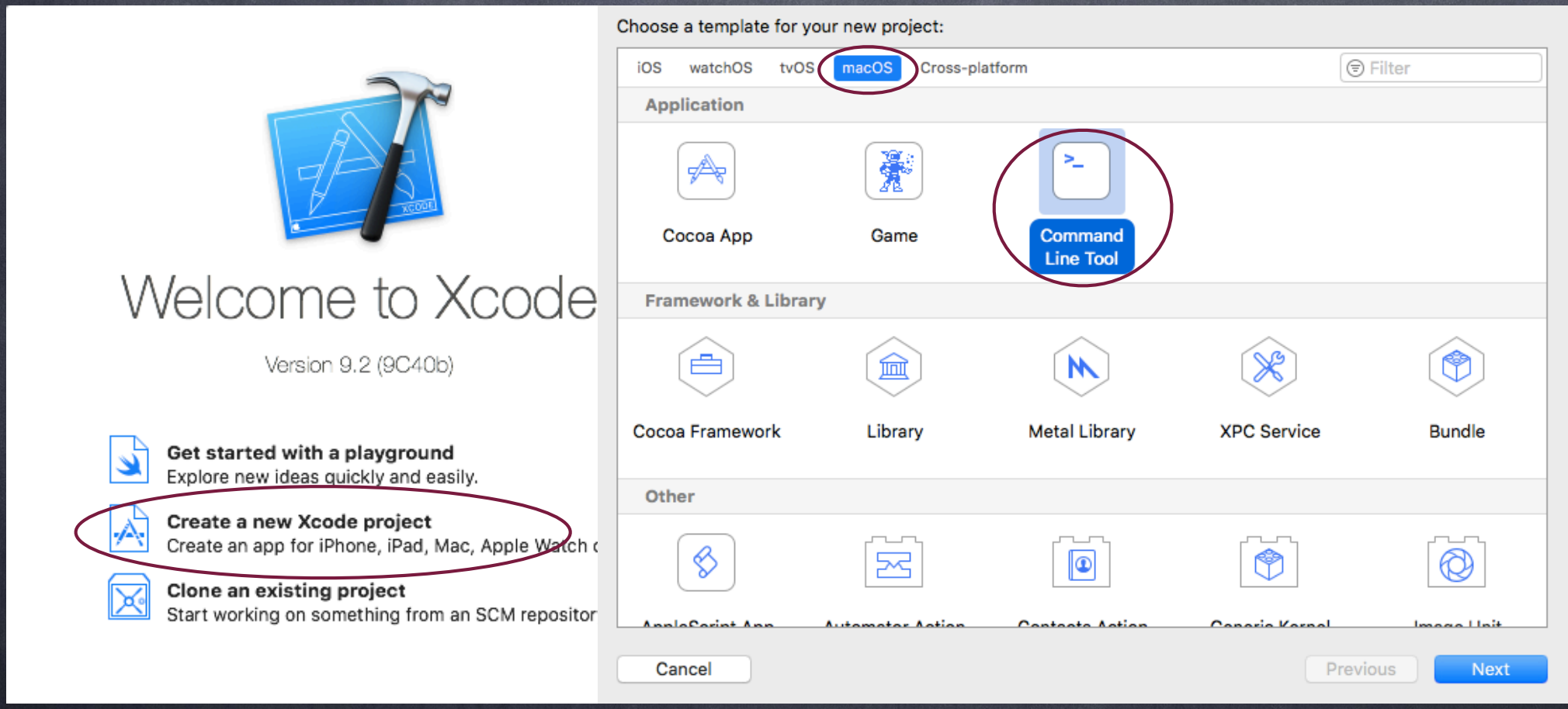
    foi: 0000007.60

Prática

Criação de um programa para testarmos todos os conceitos deste tópico - veja os próximos slides.

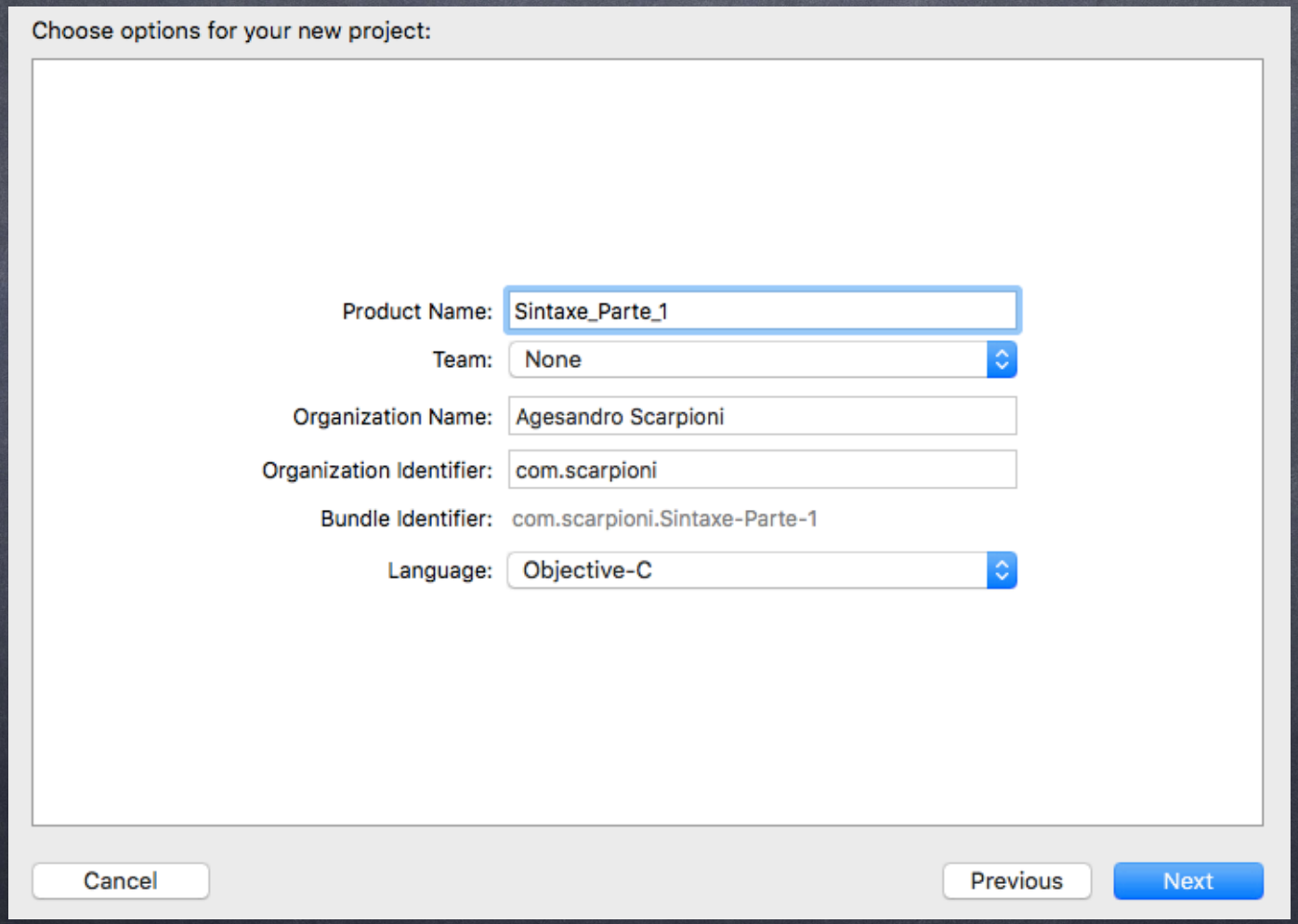
X-Code

- Abra o Xcode e clique em: Create a New Xcode Project, depois escolha macOS, selecione Command Line Tool e clique em Next.



X-Code

- Informe o nome da aplicação em Product Name, preencha o Organization Name com o nome do desenvolvedor ou empresa, em Organization Identifier utilize um nome com URL invertida, escolha a linguagem Objective-C e clique em Next.



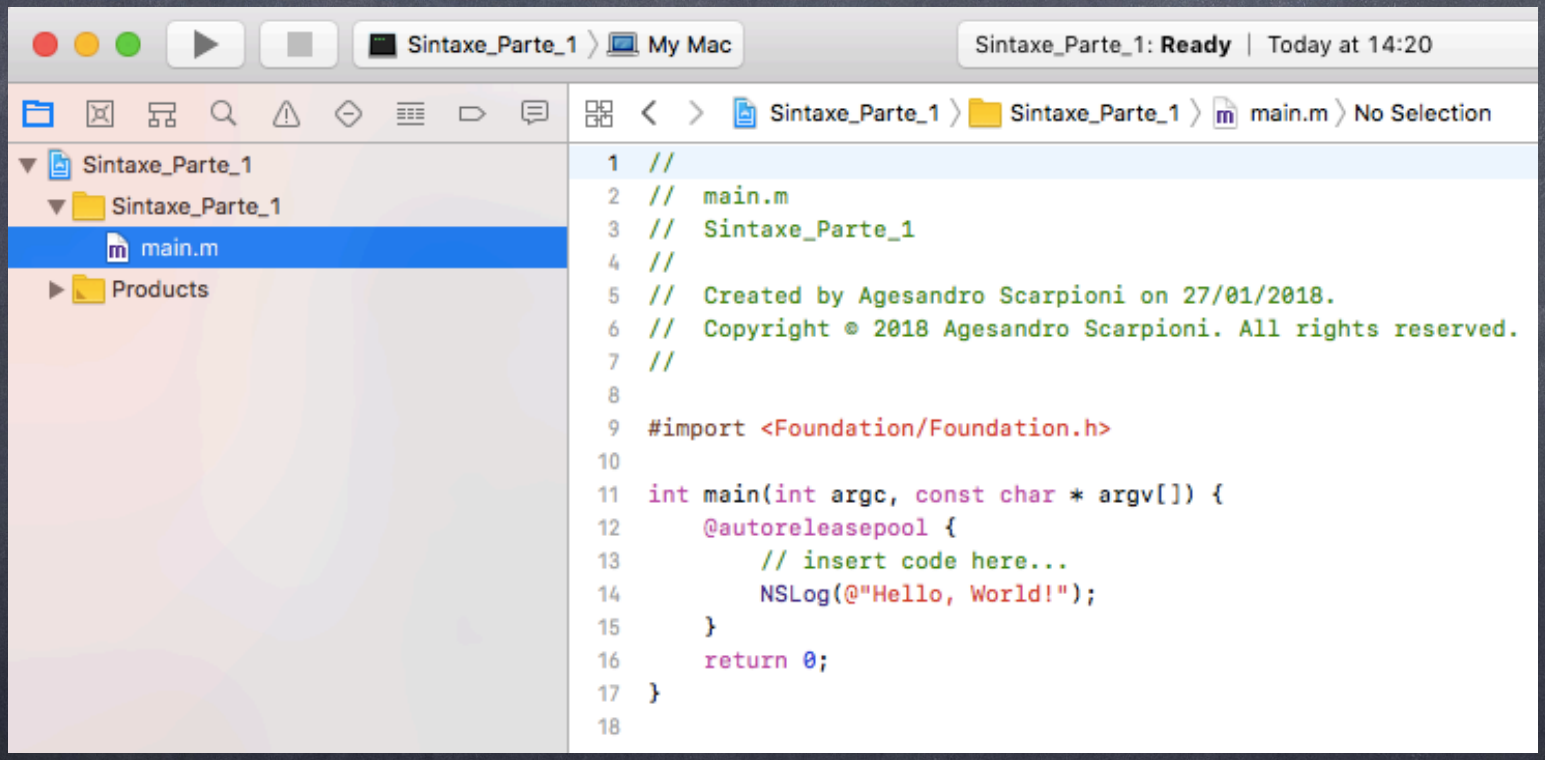
- Na tela seguinte salve na pasta padrão, clicando em Create.

OBS: O Organization Identifier é semelhante ao NameSpace no Visual Studio ou o Package no Java.

X-Code

- No canto à esquerda selecione a pasta main abaixo do nome do projeto e do lado direito REMOVA a linha: NSLog(@"Hello, Word!?);

- OK, Vamos por a mão na massa e fazer nossos testes.



X-Code - Prática

- Crie as seguintes variáveis primitivas inteiras:
  - "idade" não permita números negativos e gaste a menor quantidade de memória possível.
  - "valor" você deve inicializa-la com o valor de 1 bilhão, ou seja, esta será uma constante.
  - "totdias" deve ser inteira.
- Crie uma variável "nome" do tipo NSString.
- Atribua seu nome e sobrenome à variável nome, atribua sua idade à variável idade.
- Exiba uma mensagem "O prêmio acumulado é de <x>", onde <x> é o valor que você inicializou.
- Calcule quantos dias aproximadamente você já viveu multiplicando sua idade por 365 e exiba uma mensagem com NSLog da seguinte forma: "<nome> você já viveu aproximadamente <X> dias em <X> anos".
- Monte um if e exiba: "Você é maior" se sua idade digitada for maior ou igual a 18, caso contrário mostre: "Você é menor".
- Crie um enum com os dias da semana (DOM,SEG,TER,QUA,QUI,SEX,SAB), peça para o usuário escolher um número que represente o dia, exiba o dia da semana completo, exemplo: se o usuário digitar 6, exiba sexta-feira
- Faça um teste com o BOOL e verifique você mesmo porque é melhor perguntar usando !=YES.