

# Научно- исследовательск ая практика

---

ФУНКЦИИ, КЛЮЧЕВЫЕ СЛОВА, ЦИКЛЫ

# На прошлом занятии:

---

1. Запустили среду разработки
2. Посмотрели "Hello, world"
3. Выяснили, что какие бывают комментарии
4. Разобрали основные типы данных
5. Выяснили, как устроен цикл for
6. Написали программу для вывода чисел Фибоначчи

# Программа для чисел

```
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    int first=1, second=1;
    cout << "1 "<< first << endl;
    cout << "2 "<<second << endl;
    int next = 0;
    for (int i = 0; i< 8; i++)
    {
        next = first + second;
        cout << i+3 << " " << next << endl;
        first = second; second = next;
    }
}
```

# Области видимости

---

Можно выделить 3 различных области видимости:

- Блок
- Прототип функции
- Файл

# Области ВИДИМОСТИ

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

int someVariable;

int main(int argc, char *argv[])
{
    int anotherVariable = 1;
    cout << "Some variable " << someVariable << endl;
    cout << "Other variable " << anotherVariable << endl;
    float someVariable = 3.14;
    cout << "Some variable " << someVariable << endl;
    for (int i=0; i<2; i++)
    {
        int j = i;
        cout << "i=" << i << " j=" << j << endl;
        float anotherVariable = 2.78;
        cout << "Other variable " << anotherVariable << endl;
    }
    cout << "Other variable " << anotherVariable << endl;
}
```

# Результат

---

```
Starting /Users/amakashov/projects/build-test_2seminar-Desktop-Debug/test_2seminar...
Some variable 0
Other variable 1
Some variable 3.14
i=0 j=0
Other variable 2.78
i=1 j=1
Other variable 2.78
Other variable 1
/Users/amakashov/projects/build-test_2seminar-Desktop-Debug/test_2seminar exited with code 0
```

# Функции

---

Функции – позволяют выделить блок программы для использования

*возвращаемый\_тип* functionName

*(тип\_переменной var1, тип\_переменной var2, ...)*

- *int* Summ (*int* a, *int* b)

Принимает 2 целочисленных переменных, возвращает целочисленное значение

- *unsigned int* AbsIntValue (*float* value)

Принимает 1 значение с плавающей запятой, возвращает беззнаковое целое

- *void* PrintMessage()

Ничего не принимает и не возвращает

# ФУНКЦИИ

```
#include <iostream>

using namespace std;

const int a = 15;

int PrintFibonacci(int maxNumber);

int main()
{
    int maxNumber = 10;
    // a = 11;      // Ошибка - присваивание константной переменной
    cout << "We have global variable a=" << a << " and local variable maxNumber=" << maxNumber << endl;
    int lastNumber = PrintFibonacci(maxNumber);
    cout << "Last printed Fibonacci number is " << lastNumber << " and maxNumber is " << maxNumber << endl;
    return 0;
}

int PrintFibonacci(int maxNumber)
{
    int a=1, b=1, c;
    cout << "1 Fibonacci number " << a << endl;
    cout << "2 Fibonacci number " << b << endl;
    for (int i=2; i<maxNumber; i++)
    {
        c = a+b;
        cout << i << " Fibonacci number " << c << endl;
        a=b;
        b=c;
    }
    maxNumber = a;
    return c;
    cout << "This string should never been printed" << endl;
}
```

In function 'int main()':

❗ assignment of read-only variable 'a'



# Функции

---

```
We have global variable a=15 and local variable maxNumber=10
1 Fibonacci number 1
2 Fibonacci number 1
2 Fibonacci number 2
3 Fibonacci number 3
4 Fibonacci number 5
5 Fibonacci number 8
6 Fibonacci number 13
7 Fibonacci number 21
8 Fibonacci number 34
9 Fibonacci number 55
Last printed Fibonacci number is 55 and maxNumber is 10
Press <RETURN> to close this window...
```



# Циклы while и do while

---

Циклы с условием окончания

```
while (условие истинно)  
{  
    ...           //      Тело цикла  
}
```

Количество циклов неизвестно

Может быть бесконечным

- Вообще говоря, должно быть условие выхода

# Циклы while и do while

---

Аналогичный

```
do
{
    ...      //      Тело цикла
}
while (условие истинно)
```

Всегда выполнится хотя бы 1 раз

Может быть бесконечным

- Вообще говоря, должно быть условие выхода

# Функции

---

```
#include <iostream>

using namespace std;

void PrintMessage();    // Объявление функции PrintMessage - нужно для того, чтобы было понятно, что вызывать

int CheckOddity(int number) // Объявление И определение функции
{
    PrintMessage();        // Такую функцию нельзя вызывать "выше" по тексту
    if (number%2 == 1)      // Функция ещё не определена, но мы уже знаем её сигнатуру - список передаваемых и возвращаемых параметров
        return true;       // Вычисляем остаток от деления на 2, и если он равен 1...
    else                   // ...возвращаем true
        return false;      // иначе false
}

int main()
{
    int i=0;
    while (i<10)
    {
        if (CheckOddity(i)) // если функция вернула true
            cout << i << " is odd" << endl;
        else               // иначе
            cout << i << " is even" << endl;
        i++;
    }
    return 0;
}

void PrintMessage()      // Определение функции PrintMessage - может быть в "любом" месте
                        // И даже в другом файле!
{
    cout << "Message from PrintMessage() function" << endl;
}
```

# ФУНКЦИИ

---

```
Message from PrintMessage() function
0 is even
Message from PrintMessage() function
1 is odd
Message from PrintMessage() function
2 is even
Message from PrintMessage() function
3 is odd
Message from PrintMessage() function
4 is even
Message from PrintMessage() function
5 is odd
Message from PrintMessage() function
6 is even
Message from PrintMessage() function
7 is odd
Message from PrintMessage() function
8 is even
Message from PrintMessage() function
9 is odd
Press <RETURN> to close this window...
```

# Функции

```
#include <iostream>

using namespace std;

void PrintMessage();    // Объявление функции PrintMessage - нужно для того, чтобы было понятно, что вызывать

int CheckOddity(int number) // Объявление И определение функции
{
    PrintMessage();        // Такую функцию нельзя вызывать "выше" по тексту
    if (number%2 == 1)      // Функция ещё не определена, но мы уже знаем её сигнатуру - список передаваемых и возвращаемых параметров
        return true;       // Вычисляем остаток от деления на 2, и если он равен 1...
    else                    // ...возвращаем true
        return false;      // иначе false
}

int main()
{
    int i=0;
    // while (i<10)          // Вместо while
    for ( ; i<10; )          // используем цикл for БЕЗ начального условия и условия перехода
    {
        if (CheckOddity(i)) // Если функция вернула true
            cout << i << " is odd" << endl;
        // else              // Уберём условие else
        //     cout << i << " is even" << endl;
        i++;
    }
    return 0;
}

void PrintMessage()        // Определение функции PrintMessage - может быть в "любом" месте
                           // И даже в другом файле!
{
    cout << "Message from PrintMessage() function" << endl;
}
```

# ФУНКЦИИ

---

```
Message from PrintMessage() function
Message from PrintMessage() function
1 is odd
Message from PrintMessage() function
Message from PrintMessage() function
3 is odd
Message from PrintMessage() function
Message from PrintMessage() function
5 is odd
Message from PrintMessage() function
Message from PrintMessage() function
7 is odd
Message from PrintMessage() function
Message from PrintMessage() function
9 is odd
Press <RETURN> to close this window...
```



# Как собирается программа?

---

Препроцессинг



Ассемблирование\*



Компиляция



Линковка



# Препроцессинг

---

Обработка директив, начинающихся с #

например,

```
#define MAX_NUMBER 100
```

На этапе препроцессинга производится ЗАМЕНА директивы на её текст

В примере выше – MAX\_NUMBER будет заменен на 100

# Библиотеки и #define

---

Директива #define позволяет определить идентификатор

- #define MY\_DEFINITION – просто определяет идентификатор

```
#define LINUX_VERSION
```

```
...
```

```
#if defined(LINUX_VERSION)
```

```
    ...          //      код для Линукс-версии
```

```
#else
```

```
    ...          //      код для Windows-версии
```

```
#endif
```

#define PI 4    так можно определить константу

```
double area = r*r*PI;                    и посчитать
```

# Библиотеки и #define

---

Мы подключаем заголовочные файлы директивой

```
#include <имя_библиотеки>
```

`#include <iostream>` - стандартный заголовочный файл

`#include "myLibrary.h"` – наш собственный заголовочный файл

# Компиляция

---

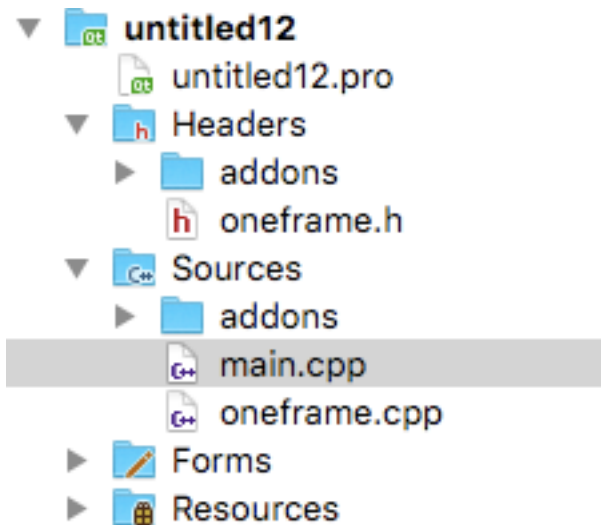
Для каждого .cpp файла, который входит в состав программы, собирается объектный файл, уже на машинном языке

В случае, если в проекте несколько файлов исходного кода – несколько объектных файлов

Для чего это, и что такое линковка?

# Состав программы

---



```
#include "onframe.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    PULT::OneFrame w;
    w.show();

    return a.exec();
}
```

# Состав программы

---

Разбиение на отдельные файлы позволяет более явно структурировать программу

Упрощает повторное использование кода

Но теперь у нас куча объектных файлов

Файлы с заголовками функций - .h, с их реализацией - .cpp

Подключение заголовочного файла

```
#include "filename.h"
```

# Линковка

---

Сборка исполняемого файла из откомпилированных объектных

Могут линковаться как ваши собственные, так и системные файлы (.dll)

# Линковка

---

```
SOURCES += main.cpp \  
          oneframe.cpp \  
          addons/mapwidget.cpp \  
          addons/Compass.cpp \  
          addons/Gyrocompass.cpp
```

```
HEADERS += \  
          oneframe.h \  
          addons/mapwidget.h \  
          addons/Compass.h \  
          addons/Gyrocompass.h
```



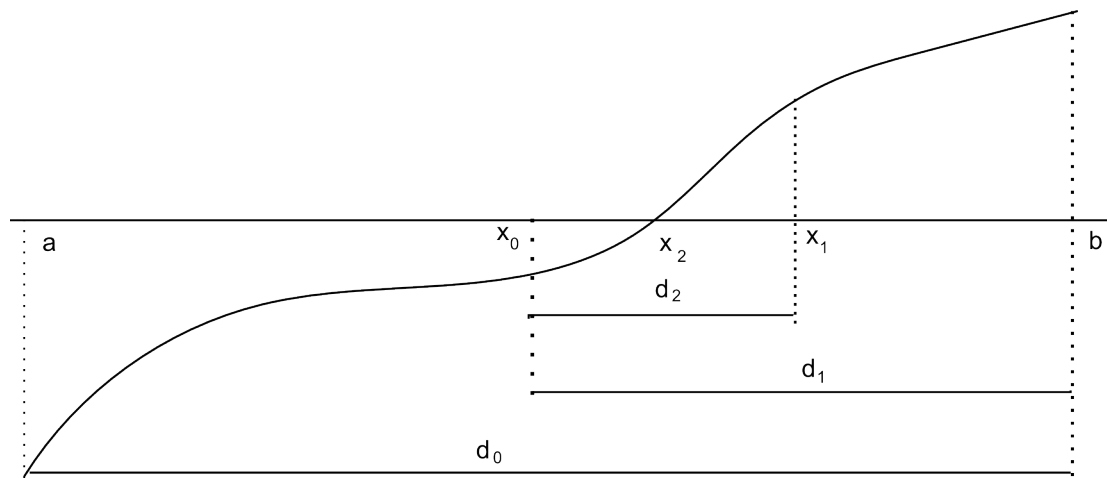
# cmath

```
1 #include <iostream>
2
3 #define _USE_MATH_DEFINES
4 #include <cmath>
5
6 using namespace std;
7
8 int main()
9 {
10     float f = -2.78;
11     int i = -4;
12     cout << "Integer variable " << i ;
13     cout << " and it's absolute value " << abs(i) << endl;
14     cout << "Floating-point variable " << f ;
15     cout << " and it's absolute value " << fabs(f) << endl;
16     float fPi = M_PI;
17     cout << "Pi=" << fPi << " pi/4=" << M_PI_4 << " and sin(pi/4)=" << sin(M_PI_4) << endl;
18     cout << "Min(f,i)=" << std::min(f,(float)i) << fixed << endl;
19     return 0;
20 }
21
22
```

```
Integer variable -4 and it's absolute value 4
Floating-point variable -2.78 and it's absolute value 2.78
Pi=3.14159 pi/4=0.785398 and sin(pi/4)=0.707107
Min(f,i)=-4
Press <RETURN> to close this window...
```

# Метод половинного деления

---



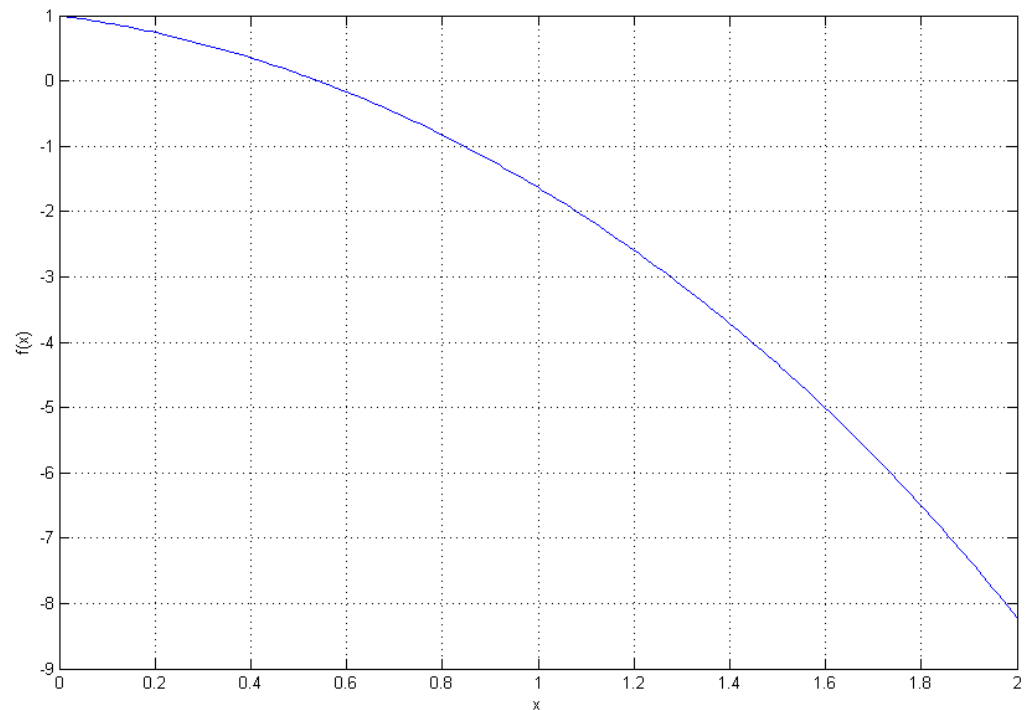
# Метод половинного деления

---

Давайте решим следующую задачу:

- интервал  $x \in [0 \dots 2]$
- Функция

$$f(x) = -e^x + 2\cos x$$



# Метод половинного деления

---

```
#include <iostream>
#define USE_MATH_DEFINES
#include <cmath>

using namespace std;

double function (double x);

int main()
{
    const double eps = 0.01;
    float leftBound = 0,
          rightBound = 2;
    if (function(leftBound)*function(rightBound)<0)
    {
        double midX;
        do
        {
            // Вставьте сюда свой код!
        }
        while(fabs(function(midX))>eps);
    }
    else
    {
        cout << "Unable to find root on interval [" << leftBound << ", "<<rightBound<<"]\n";
    }
    return 0;
}

double function (double x)
{
    return (-exp(x)+ 2 * cos(x));
}
```

# Метод половинного деления

```
#include <iostream>
#define USE_MATH_DEFINES
#include <cmath>

using namespace std;

double function (double x);

int main()
{
    const double eps = 0.01;
    float leftBound = 0,
          rightBound = 2;
    if (function(leftBound)*function(rightBound)<0)
    {
        double midX;
        do
        {
            midX = (leftBound+rightBound)/2;
            cout << std::fixed;
            cout << "We using interval [" << leftBound << "," << rightBound << "]" << endl;
            cout << "Midddle point is " << midX << " and function value " << function(midX) << endl << endl;
            if (function(midX)*function(leftBound)<0)
                rightBound = midX;
            else
                leftBound = midX;
        }
        while(fabs(function(midX))>eps);
    }
    else
    {
        cout << "Unable to find root on interval [" << leftBound << "," << rightBound << "]\n";
    }
    return 0;
}

double function (double x)
{
    return (-exp(x)+ 2 * cos(x));
}
```

# Метод половинного деления

---

```
We using interval [0.000000,2.000000]
Middle point is 1.000000 and function value -1.637677

We using interval [0.000000,1.000000]
Middle point is 0.500000 and function value 0.106444

We using interval [0.500000,1.000000]
Middle point is 0.750000 and function value -0.653622

We using interval [0.500000,0.750000]
Middle point is 0.625000 and function value -0.246320

We using interval [0.500000,0.625000]
Middle point is 0.562500 and function value -0.063206

We using interval [0.500000,0.562500]
Middle point is 0.531250 and function value 0.023292

We using interval [0.531250,0.562500]
Middle point is 0.546875 and function value -0.019538

We using interval [0.531250,0.546875]
Middle point is 0.539062 and function value 0.001982

Press <RETURN> to close this window...
```

# Логические операторы

---

Нужно одновременное выполнение нескольких условий

- `(условие_1) && (условие_2)` – логическое И
- `(условие_1) || (условие_2)` – логическое ИЛИ
- `(условие_1) && !(условие_2)` - отрицание

# Метод половинного деления

```
#include <iostream>
#define USE_MATH_DEFINES
#include <cmath>

using namespace std;

double function (double x);

int main()
{
    const double eps = 0.00000001;
    float leftBound = 0,
          rightBound = 2;
    if (function(leftBound)*function(rightBound)<0)
    {
        double midX;
        int iteration = 1;
        do
        {
            midX = (leftBound+rightBound)/2;
            cout << std::fixed;
            cout << "Iteration " << iteration << " we using interval [" << leftBound << "," << rightBound << "]" << endl;
            cout << "Middlle point is " << midX << " and function value " << function(midX) << endl<< endl;
            if (function(midX)*function(leftBound)<0)
                rightBound = midX;
            else
                leftBound = midX;
            iteration++;
        }
        while(fabs(function(midX))>eps && iteration<50);
    }
    else
    {
        cout << "Unable to find root on interval [" << leftBound << "," << rightBound << "]\n";
    }
    return 0;
}

double function (double x)
{
    return (-exp(x)+ 2 * cos(x));
}
```



# Метод половинного деления

---

```
Iteration 43 we using interval [0.539785,0.539785]  
Middle point is 0.539785 and function value 0.000000
```

```
Iteration 44 we using interval [0.539785,0.539785]  
Middle point is 0.539785 and function value 0.000000
```

```
Iteration 45 we using interval [0.539785,0.539785]  
Middle point is 0.539785 and function value 0.000000
```

```
Iteration 46 we using interval [0.539785,0.539785]  
Middle point is 0.539785 and function value 0.000000
```

```
Iteration 47 we using interval [0.539785,0.539785]  
Middle point is 0.539785 and function value 0.000000
```

```
Iteration 48 we using interval [0.539785,0.539785]  
Middle point is 0.539785 and function value 0.000000
```

```
Iteration 49 we using interval [0.539785,0.539785]  
Middle point is 0.539785 and function value 0.000000
```

```
Press <RETURN> to close this window...
```