Основы C++

День 1.

Небольшой экскурс в историю вычислительных устройств. Разностная машина Бэббиджа, вклад Ады Лавлейс в развитие понятия программирования. Развитие электронных логических элементов и устройств, понятие алгоритма. Сложности создания алгоритмов на заре появления ЭВМ, отладка и накопление ошибок как краеугольный камень программирования на низком уровне.

Язык программирования как средство оцифровывания алгоритмов. Языки ассемблера и его эволюция в языки высокого уровня. Понятие компилятора трансляция конструкций языков высокого уровня в машинные команды.

Первоначальные сведения о языках C и C++, достоинства и недостатки.

Написание первой программы. Запуск MinGW, настройка проекта. Несколько начальных слов о структуре программы/проекта C++ -- файлы исходного кода, хэдэры, библиотеки. Main.cpp:

```
#include <iostream.h>

int main(){
 printf("Hello, world!\n");
 return 0;
}
```

Задание: вывод технических символов из таблицы ASCII (звонок, перевод каретки и пр.)

День 2.

Стандартные типы данных и место в памяти. Переменные – участки памяти, которые можно читать и менять. Различаются переменные по занимаемому месту в памяти и возможным манипуляциям.

bool – булевое true/false, **1 байт**

char – целое знаковое, $(-2^8+1)..(2^8)$, 1 байт

int – целое знаковое, $(-2^{32}+1)..(2^{32})$, 4 байта

float – с плавающей точкой, 8 байт

double – двойной точности, **10 байт**

Приведение типов – к высшему. Операции и очередность, примеры (+-*/=%++--).

Перед использованием – объявление переменной:

```
char c = 'A'; // c = 65
int i = 2;
float f;
f = 3.5f
f = 5e-12;
double pi = 3.14159265358;
f = pi;
i = c;
printf("%c %d %f %lf \n", c, i, f, pi);
i = 65;
printf("%c \n", i);
```

Примеры с арифметическими действиями.

Функция printf – тип и количество параметров, способы вывода, интерпретация в зависимости от типа параметра и спецификатора вывода.
Задание: Научиться выводить любые ASCII символы.

Несколько слов о структурах и классах.

```
class Complex{
 double re, im;

 Complex(){
  re = im = 0;
 }

 Complex(double RE, double IM){
  re = RE;
  im = IM;
 }
};
//////////////////////
Complex Z1;
Complex Z2(-2, 4e-4);
```

Информация о конструкторах, деструкторах, функциях-членах класса.

День 3.

Управляющие структуры (if-else, do-while, for, switch, break, continue)

Способ ввода с клавиатуры с помощью scanf или cin. Пример: программа с бесконечным циклом, ожиданием ввода и ответом на действия пользователя.

Написание собственных функций, синтаксис объявления. Примеры простейших функций: сложение числа, вывод сообщений количеством введенным с клавиатуры. Немного о типе передаваемых параметров и возвращаемых значениях. Рекурсия, использование внутри функции самой себя. Примеры: Факториал, числа Фибоначчи. Плюсы и минусы рекурсивного подхода. Итерационный вид тех же функций.

Математические функции из math.h.

День 4.

Указатель – всего лишь **переменная, хранящая адрес другой переменной**. Тип указателя, операция разыменования, получения адреса другой переменной.

```
int i = 20;
int *ptr; // (int *) ptr;
ptr = &i;
*ptr = 10;
printf("%d\n", i);

////////

void Inc(int *ptr){
 *ptr++;
}

////////

Inc(&i);
printf("%d\n", i);
```

Детальный разбор примера.

Передача параметров по ссылке и о множественное возвращаемое значение.

Массивы – блоки переменных одного типа, расположенные в памяти линейно, и имеющие определенные правила доступа. Пример:

```
int arr[10];
arr[0] = 1;
arr[1] = arr[2] = 23;

printf("%d %d %d\n", arr[0], arr[2], arr[3]);

int *ptr = arr;
ptr[0] = 666;
printf("%d\n", arr[0]);
```

Многомерные массивы и массивы указателей. Пример:

```
int arr2D[2][3];
arr2D[0][0] = 777;
arr2D[1][2] = arr[0][0];

int **ptr2 = arr2D;
int *ptr1 = arr2D[1];

printf("%d %d %d\n", arr[1][2], ptr2[1][2], ptr1[2]);
// (ptr1)[2] == (arr2D[1])[2] == ptr2[1][2]
```

5 день.

Динамическое выделение памяти. Функции malloc(), calloc(), free() в C и операторы new, delete и delete[] в C++. Применение динамической памяти в случае, если заранее не известен размер задачи.

Пример:

```
int N = 10; // не константа!

int *dynC, *dynCPP;

dynC = (int *)malloc(N*sizeof(int));
dynCPP = new int[N];

dynC[0] = dynCPP[0] = 777;

printf("%d %d\n", dynC[0], dynCPP[0]);

free(dynC);
delete[] dynCPP;
```

Случай работы с многомерными массивами:

```
int N = 4, M = 2;

int **dynC, **dynCPP;

dynC = (int **)malloc(N*sizeof(int *));
for(int i=0; i<N; i++)
 dynC[i] = (int *)malloc(M*sizeof(int));

dynCPP = new int*[N];
for(int i=0; i<N; i++)
 dynCPP[i] = new int[M];

dynC[0][0] = dynCPP[0][0] = 777;

printf("%d %d\n", dynC[0][0], dynCPP[0][0]);

for(int i=0; i<N; i++)
 free(dynC[i])
free(dynC);

for(int i=0; i<N; i++)
 delete[] dynCPP[i];
delete[] dynCPP;
```

Дополнительный материал: stl.