# 3.1 Массивы

# Синтаксис:

<тип данных> <имя массива>[число элементов];

**Например:**

int array_int[100]; //одномерный массив 100 целочисленных элементов

double array_d[25]; //одномерный массив 25 вещественных элементов

## Пример программы:

Запись в массив значений линейной функции f(x)=kx+b

```
double k=0.5, b = 10.0;
double f[100];
for(int x=0;i < 100;i++)
{
f[i] = k*x+b;
printf(“%.2f ”,f[i]);
}
```

# Инициализация массива в момент его объявления

Пример 2(инициализация массива в момент его объявления):

```
int powers[4] = {1, 2, 4, 6};
```

Пример 3 (программа приведет к ошибке при компиляции):

```
int N=100;
float array_f[N]; //ошибка, так нельзя
```

# При объявлении массивов используют подход:

```
#include
#define N 100
int main()
{
float array_f[N];
return 0;
}
```

# Пример, когда число элементов при инициализации будет меньше размерности массива:

```
#define SIZE 4
int data[SIZE]={512, 1024};
for(int i = 0;i < SIZE;i++)
printf(“%d, \n”,data[i]);
```

Результатом работы программы будет:

512, 1024, 0, 0

Когда наперед неизвестно число элементов, целесообразно использовать такую конструкцию языка C++:

```
int data[] = {2, 16, 32, 64, 128, 256};
```

# Объявление двумерных массивов:

int array2D[100][20]; //двумерный массив 100x20 элементов

array2D[0][0]

array2D[0][1]

и т.д.

# Для начальной инициализации двумерного массива

- long array2D[3][2] = {{1, 2}, {3, 4}, {5, 6}};

или

- long array2D[][] = {{1, 2}, {3, 4}, {5, 6}};

# 3.2. Работа со строками

Пример 1 (использование строк в программе)

```
char str_1[100] = {‘П’,’р’,’и’,’в’,’е’,’т’,’\0’};
char str_2[100] = “Привет”;
char str_3[] = “Привет”;
printf(“%s\n%s\n%s\n”,str_1,str_2,str_3);
```

# Пример 2: особенность использования специального символа ‘\0’

```
char str1[10] = {‘Л’,’е’,’к’,’ц’,’и’,’я’,’\0’};
char str2[10] = {‘Л’,’е’,’к’,’ц’, ’\0’,’и’,’я’ };
char str3[10] = {‘Л’,’е’, ’\0’,’к’,’ц’,’и’,’я’ };
printf(“%s\n%s\n%s\n”,str1,str2,str3);
```

Результат работы программы:

Лекция
Лекц
Ле

# Пример 3. Программа вычисления длины строки.

```
#include
int main(void)
{
char str[] = “Привет мир!”;
int size_array = sizeof(str);
int length = 0;
while(length < size_array && str[length] != ‘\0’) length++;
printf(“Длина строки = %d.\n”,length);

return 0;
}
```

Функция вычисления размера строк уже реализована в стандартной библиотеке языка C++ string.h со следующим синтаксисом:

- int strlen(char* str);

где char* str – указатель на строку

# Правило использования функции strlen()

Пример 4. Пример использования функции strlen().

```
#include
#include
int main(void) {
char str[] = “Привет мир!”;
int length = strlen(str);
printf(“Длина строки = %d.\n”,length);
return 0;
}
```

Результатом работы программы будет вывод на экран числа 11.

# Правила присваивания одной строковой переменной другой

Допустим объявлены две строки

char str1[] = “Это первая строка”;
char str2[] = “Это вторая строка”;

Необходимо выполнить оператор присваивания str1 = str2;

**Необходимо перебирать по порядку элементы одного массива и присваивать их другому массиву**

Пример 5.

```
char str1[] = “Это первая строка”;
char str2[] = “Это вторая строка”;
int size_array = sizeof(str1);
int i=0;
while(i < size_array && str1[i] != ‘\0’) {
str2[i] = str1[i];
i++;
}
str2[i] = ‘\0’;
```

# Реализация в библиотеке C++

Библиотека string.h

Определение:

- char* strcpy (char* dest, char* src);