**Введение**

**Языки программирования**

**Машинный язык** – «естественный».

**BB 11 01 B9 0D 00 B4 0E 8A 07 43 CD 10 E2 F9 CD 20 48 65 6C 6C 6F 2C 20 57 6F 72 6C 64 21**

**Ассемблерный язык** – для элементарных операций англоязычные аббревиатуры.

**mov bx, 0111h**

**mov cx, 000Dh**

**mov ah, 0Eh**

**mov al, [bx]**

**inc bx**

**int 10h**

**loop 0108**

**int 20h**

**HW db 'Hello, World!'**

**Языки высокого уровня** – для сложных действий операторы.

**Лабораторные работы**

**Система C/C++**

1. **Среда разработки программ** (Integrated Development Environment, IDE) **Microsoft Visual Studio**

2. **Язык** **C/C++**

3. **Стандартная библиотека** **С/C++**

**1. Создание нового проекта.**

File 

New 

Project 

Visual C++ 

Win32 Win32 Consol Application

**Требования к выполнению лабораторных работ**

Имена проектов должны содержать фамилию разработчика, набранную английскими буквами, и номер лабораторной работы: **Bogachenko\_1**.

Application Settings Empty project

**2. Добавление нового файла.**

Project 

Add New Item

C++ File (.cpp)

**3. Компиляция.**

Build 

Build Solution (F7)

**4. Выполнение.**

Debug 

Start Without Debugging (Ctrl + F5)

**5. Отладка.**

F5;

F10 / F11 (с заходом в функции)

**Стадии программы на языке C/C++**

1. Редактирование исходного кода: **.c** или **.cpp**

2. Препроцессорная обработка: **.h**

3. Компиляция: **.obj**

4. Компоновка (linking): **.exe**

5. Загрузка

6. Исполнение

**Глава I. Структурное программирование**

* 1. **Первая программа на C/C++**

|  |  |
| --- | --- |
| Функции форматированного  ввода/вывода **printf()** и **scanf()**.  **//Программа в стиле С**  **#include <stdio.h>**  **int main() {**  **printf("Welcome to C++!\n");**  **return 0;**  **}** | Ввод/вывод при помощи объектов  **cin** и **cout**.  **//Программа в стиле С++**  **#include <iostream.h>**  **int main(){**  **std::cout << "Welcome to C++!\n";**  **return 0;**  **}** |

- комментарий (одно- и многострочный)

- директива препроцессору

- обязательная функция, возвращаемое значение, «тело» функции

- оператор: простой (**;**) и составной(**{}**)

- пространство имен

- операция передачи в поток

- переход на новую строку

- выход из функции

1. **Вторая программа на C/C++**

**//Программа сложения двух целых чисел;**

**//ввод/вывод с помощью объектов в стиле C++**

**#include <iostream>**

**int main() {**

**int num1;**

**int num2;**

**int sum;**

**std::cout << "Enter first integer: ";**

**std::cin >> num1;**

**std::cout << "Enter second integer: ";**

**std::cin >> num2;**

**sum = num1 + num2;**

**std::cout << "Sum is " << sum << std::endl;**

**return 0;**

**}**

- операция извлечения из потока

- манипулятор потока

- операция присваивания

**// Программа сложения двух целых чисел;**

**// ввод/вывод с помощью функций в стиле С**

**#include <stdio.h>**

**int main() {**

**int num1;**

**int num2;**

**int sum;**

**printf("Enter first integer: ");**

**scanf("%i", &num1);**

**printf("Enter second integer: ");**

**scanf("%i", &num2);**

**sum = num1 + num2;**

**printf("Sum is %i\n", sum);**

**return 0;**

**}**

- спецификации преобразования

- операция получения адреса

- в обоих вариантах нет проверки корректности входных данных – нет «защиты от дурака».

**setlocale(LC\_ALL,"Russian");**

**system("Pause");**

1. **Переменные**

имя тип – размер значение

**3.1. Идентификаторы (имена)**

1. Идентификатор – имя

2. S\_1 ≠ s\_1

3. iMaxLength или max\_length

4. Длина

5. ≠ *ключевые слова* ([1], с.19)

**asm**

**auto**

**bool**

**break**

**case**

**catch**

**char**

**class**

**const**

**const\_cast**

**continue**

**default**

**delete**

**do**

**double**

**dynamic\_cast**

**else**

**enum**

**explicit**

**export**

**extern**

**false**

**float**

**for**

**friend**

**goto**

**if**

**inline**

**int**

**long**

**mutable**

**namespace**

**new**

**operator**

**private**

**protected**

**public**

**register**

**reinterpret\_cast**

**return**

**short**

**signed**

**sizeof**

**static**

**static\_cast**

**struct**

**switch**

**template**

**this**

**throw**

**true**

**try**

**typedef**

**typeid**

**typename**

**union**

**unsigned**

**using**

**virtual**

**void**

**volatile**

**wchar\_t**

**while**

* + 1. **Типы данных**

*Чем отличаются данные различных типов?*

1. Представление.

2. Значения.

3. Действия.

*Типы данных:*

1. Стандартные (основные, встроенные).

2. Составные

***Стандартные типы***

• **int** (целый);

• **char** (символьный);

целочисленные

• **wchar\_t** (расширенный символьный) (C++);

• **bool** (логический) (C++);

• **float** (вещественный);

вещественные (с плавающей точкой)

• **double** (вещ. с двойной точностью).

*Спецификаторы типа*:

• **short** (короткий);

• **long** (длинный);

• **signed** (знаковый);

• **unsigned** (беззнаковый).

Уточняют внутреннее представление и диапазон значений стандартных типов.

***Целый тип***

*Внутреннее представление* — целое число в двоичном коде.

**[signed] int** – 4 байта

sizeof(int)

**short [int]** – 2 байта

**long [int]** – 4 байта

**signed [int]** – 4 байта (старший бит – знак числа)

**unsigned [int]** – 4 байта

|  |  |  |  |
| --- | --- | --- | --- |
| *Тип* | *Диапазон значений* | *Размер (байт)* |  |
| **signed short int** | -32 768 ... 32 767 | 2 | 215-1 |
| **unsigned short int** | 0 ... 65 535 | 2 | 216-1 |
| **signed long int** | -2 147 483 648 ... 2 147 483 647 | 4 |  |
| **unsigned long int** | 0 ... 4 294 967 295 | 4 |  |

***Символьный тип***

*Внутреннее представление* — целое число в двоичном коде.

**[signed] char** – 1 байт.

***Вещественные типы (с плавающей точкой)***

*Внутреннее представление:* мантисса и порядок.

**float** – 4 байта

(1 бит под знак + 8 бит под порядок + 23 бита под мантиссу)

**double** – 8 байт (по умолчанию для константных значений)

**long double** – 10 байт

***Тип void***

Множество значений 

* + 1. **Константы const**

**Константа** – переменная, значение которой менять нельзя.

**const int i = 1;**

**Константные значения:**

1. Целые: 234 -43

**0**23

**0x**2A **0X**FB2

2. Вещественные: 5.6 5. .4

0.6**e**-4 .22**E**10

3. Символьные: 's' 'R'

'\n' – нет графич. изобр.

'\\' '\"' '\''

'\**0**13'

**char i = '**\n**';**

**printf("%i",i);**

4. Строковые: "\"C++\"!\n"

|"A"|=2 '\0' – символ конца строки

**Описание переменных**

[класс памяти] [**const**] тип имя [**=** нач.значение | **(**нач.значение**)**]

Если тип начального значения не совпадает с типом переменной, то происходит **автоматическое (неявное) преобразование типов** (от меньшего размера к большему) **int i =** '\n'**;**

**Oбласть действия** – часть программы, в которой имя переменной можно использовать для доступа к связанной с ним области памяти.

*Локальная переменная* – описана и действует внутри блока {}

*Глобальная переменная* – описана вне блока, действует до конца файла

**Время жизни** – промежуток с момента выделения памяти до момента освобождения памяти.

*Постоянное время жизни* – в течение выполнения программы

*Временное время жизни* – в течение выполнения блока

**Область видимости = область действия** (но не всегда)

**int i = 1;**

**#include<stdio.h>**

**int main() {**

**int i = 2;**

**printf("%i, %i\n",i,::i);**

**return 0;**

**}**

**Классы памяти**:

**auto** временное время жизни (по умолчанию для локальных переменных, для глобальных не используется)

**static** постоянное время жизни

**register** аналог auto, но память по возможности выделяется в   
регистрах процессора

**extern** определение переменной **≠** объявление переменной

**int i; // глобальная переменная**

**#include<stdio.h>**

**int main() {**

**int i = 2, j = 3;**

**const int k = 4; // константа**

**extern int m; // объявление переменной m**

**::i = 6;**

**printf("i=%i; j=%i; k=%i; m=%i; ::i=%i\n"**

**,i,j,k,m,::i);**

**// на экране: i=2; j=3; k=4; m=5; ::i=6**

**return 0;**

**}**

**int m = 5; // определение переменной m**

*Переменная должна иметь по возможности* ***минимальную*** *область действия!*

1. **Операции**

*Арифметические операции*

**++ ––** (инкремент и декремент).В чем отличие: **++x** и **x++** ?

**/**

**int i;**

**i = 7 / 2; // i == 3**

**%** (только с целочисленными операндами)

**\***

Возведение в степень?

*Логические операции*

**== != <= < >= >**

**|| &&**

*Поразрядные операции*

**| & ^** **<<** **>>** (только с цел. операндами)

*Условная операция*

**max = (b > a) ? b : a;**

*Арифметические операции с присваиванием*

**=**

**+= -=** **\*=**  **/= %=**

- Приоритеты и ассоциативность

**12 / 6 / 2 a = b = c;**

- Преобразование типов

- Вычисление значения логического выражения происходит

ДО ПОЛУЧЕНИЯ РЕЗУЛЬТАТА

**int a = 0;**

**int b;**

**b = (a < 3)||(++a == 1);**

**printf("%i %i\n",b,a); // 1 0**

**b = (a < 3)&&(++a == 1);**

**printf("%i %i\n",b,a); // 1 1**

**int a = 1;**

**int b = 5;**

**double x = a / b; // x == 0**

**double y = ((double)a) / b; // y == 0.5**

1. **Базовые конструкции структурного программирования**

Программы в стиле «поток сознания».

Базовые конструкции структурного программирования.

**Операторы**

- простые

- составные

* 1. **Операторы принятия решения (операторы ветвления)**

Тип – любой стандартный

***Условный оператор* if**

**if** (выражение) оператор\_1 [**else** оператор\_2]

**if (a < b) {**

**if (a < c)**

**m = a;**

**else**

**m = c;**

**}**

**else {**

**if (b < c)**

**m = b;**

**else**

**m = c;**

**}**

else относится к ближайшему if

**if (a < b)**

**if (a < c)**

**m = a;**

**else**

**m = c;**

**else**

**if (b < c)**

**m = b;**

**else**

**m = c;**

***Оператор выбора* switch**

Тип - целочисленный

**switch** (выражение)**{**

**case** константное\_выражение\_1**:** [список\_операторов\_1]

**case** константное\_выражение\_2**:** [список\_операторов \_2]

…

**case** константное\_выражение\_n**:** [список\_операторов \_n]

[**default:** список\_операторов]

**}**

**#include<stdio.h>**

**int main()**

**{**

**char c;**

**printf("Enter your simbol: ");**

**scanf("%c",&c);**

**switch (c){**

**case '\*':**

**case '/':**

**case '+':**

**case '-': … break;**

**default: … // может отсутствовать**

**}**

**}**

* 1. **Операторы цикла**

Итерация – это один проход цикла

Зачем 3 конструкции оператора цикла? – для удобства!

***Цикл с предусловием* while**

Тип – любой стандартный

**while** (выражение) оператор

Тип – любой стандартный

***Цикл с постусловием* do while**

**do** оператор **while**  (выражение)

***Цикл с параметром* for**

**for** (инициализация**;** выражение**;** модификатор) оператор

Тип – любой стандартный

**int i;**

**for(i=1, int j=1; i<10; i++, j += 2){…}**

* 1. **Операторы передачи управления**

Программируем без **goto**!

**break**

**continue**

**return** [выражение]

**int main{ return 'q';}**

**printf("Press x (0 < x < 10): ");**

**scanf("%i",&x);**

**while((x <= 0)||(x >= 10)){**

**printf("Error! Press x (0 < x < 10): ");**

**scanf("%i",&x);**

**}**

или

**while(1){**

**// бесконечный цикл с принудительным выходом**

**printf("Press x (0 < x < 10): ");**

**scanf("%i",&x);**

**if ((x > 0)&&(x < 10))**

**break;**

**else**

**printf("Error! ");**

**}**

* 1. **Пример «Таблица значений функции»**

**double x1, x2, dx;**

**printf("Enter x1: ");**

**scanf("%lf",&x1);**

**printf("Enter x2: ");**

**scanf("%lf",&x2);**

**printf("Enter dx: ");**

**scanf("%lf",&dx);**

**int const lines = 3;**

**int l;**

**double x = x1, f;**

**while (x <= x2){**

**printf("-----------------------------**\n**");**

**printf("| x | f |**\n**");**

**printf("-----------------------------**\n**");**

**for (l=1; ((x <= x2)&&(l <= lines)); l++, x += dx){**

**if (x > 0){**

**f = 1/x;**

**printf("| %11.3lf | %11.3lf |**\n**",x,f);**

**} else{**

**printf("| %11.3lf | error |**\n**",x);**

**}**

**}**

**printf("-----------------------------**\n**");**

**if (x <= x2){**

**printf("Press ENTER for continue: ");**

**if (getchar() != '**\n**')**

**break;**

**}**

**}**

1. **Указатели и ссылки**

**6.1. Определение**

Переменная – значение

Указатель – адрес памяти – конкретный тип

1. Указатель на объект.

**тип \*имя;** или **тип\* имя;** или **тип \* имя;**

**int \*a, b;**

**int \*a, \*b;**

2\*. Указатель на **void**.

**void \*p;**

3\*. Указатель на функцию.

**int (\*fun)(double, double);**

Ссылка С++

– синоним имени

– не занимает пространство в памяти (в отличие от указателя)

**int c = 2;**

**int& s = c; или int &s = c;**

**s = 5;**

**printf("%i",c);**

**6.2. Инициализация указателей**

***Объявил указатель – инициализируй! Как?***

**1.** Адрес существующего объекта.

**1.1.** Операция получения адреса.

**int i = 2;**

**int\* pi = &i;**

|  |  |  |
| --- | --- | --- |
| **pi** |  | **i** |
| 0xF9100000 |  | 2 |
| ячейка 0xB8000000 |  | ячейка 0xF9100000 |

**1.2.** Другой указатель.

**int\* pi1 = pi;**

**//int\* pi1; pi1 = pi;**

|  |
| --- |
| **pi1** |
| 0xF9100000 |

**2.** Адрес в явном виде. ***Вопрос «Зачем?»***

**3.** Пустое значение. ***Хороший тон!***

**int\* pi = 0; //NULL**

**4.** Адрес выделенного участка динамической памяти (кучи).

*- динамическая переменная*

*- переменная-указатель*

*- два способа «мусорить»* ***НЕ ОСТАВЛЯЙТЕ «МУСОР»!***

|  |
| --- |
| С++  ключевые слова (операторы) |
| ***Выделение памяти*** |
| **int\* pi = new int;** |
| ***Освобождение памяти*** |
| **delete pi;** |

|  |
| --- |
| С  функции, **<malloc.h>** |
| ***Выделение памяти*** |
| **int\* pi = (int \*)malloc(sizeof(int));** |
| ***Освобождение памяти*** |
| **free(pi);** |

**6.3. Операции с указателями**

1. Получение адреса **&**
2. Разадресация или разименование **\***

**int c = 5;**

**int \*pc = 0;**

**pc = &c;**

**printf("%i",\*pc);**

**int c = 2;**

**int \*pc = new int;**

**\*pc = 5;**

**c = \*pc;**

**printf("%i",c);**

**3.** Кое-какие арифметические опрации.

1. **Массивы – последовательность однотипных величин**

**7.1. Статические массивы**

**тип имя[размерность];**

**int m\_1[10]; //инициализация не происходит**

**int m\_2[] = {3,2,1}; //размерность 3**

**int m\_3[10] = {3,2,1}; //остальные эл-ты равны 0**

**m\_1[0] = 1; //нумерация с НУЛЯ**

**// m\_1[10] = 1; - ошибка выполнения**

**// синтаксические ошибки отслеживает компилятор**

**// логические ошибки – самые страшные**

**// грамотно так:**

**const int size\_m = 10;**

**int m[size\_m] = {0};**

**int i;**

**for(i=0; i<size\_m; i++){**

**// если i<=size\_m, то ошибка выполнения**

**printf("%i",m[i]);**

**}**

**m  &m[0] m[i]  \*(m+i)**

**//имя массива – это константный указатель на его //нулевой элемент**

***Многомерные массивы –*** *располагаются в памяти построчно*

**const int n = 3, m = 2;**

**int mas\_1[n][m] = {1, 2, 3, 4, 5, 6};**

**// число строк 3, число столбцов 2**

**int mas\_2[n][m]={{1,2},{3,4},{5,6}};**

**7.2. Динамические массивы - объем памяти определяется**

**НА ЭТАПЕ ВЫПОЛНЕНИЯ ПРОГРАММЫ**

*Одномерный динамический массив*

|  |
| --- |
| С++ |
| ***Выделение памяти*** |
| **int\* massiv = new int [100];**  **//выделяется память под 100 величин типа int** |
| ***Освобождение памяти*** |
| **delete [] massiv;** |

|  |
| --- |
| С |
| ***Выделение памяти*** |
| **int\* massiv = (int\*)malloc(100\*sizeof(int));**  **//выделяется память под 100 величин типа int** |
| ***Освобождение памяти*** |
| **free(massiv);** |

*Двумерный динамический массив*

|  |
| --- |
| С++ |
| **int n; // число строк**  **int m; // число столбцов**  **int\*\* matr = new int\* [n];**  **// выделяется память под массив указателей**  **// на строки массива**  **for(int i=0; i<n; i++){**  **matr[i] = new int [m];**  **// выделяется память под m элементов типа int**  **}**  **for(int i=0; i<n; i++){**  **delete [] matr[i];**  **}**  **delete [] matr;** |

|  |
| --- |
| C |
| **int n; // число строк**  **int m; // число столбцов**  **int\*\* matr = (int\*\*)malloc(n\*sizeof(int\*));**  **// выделяется память под массив указателей**  **// на строки массива**  **for(int i=0; i<n; i++){**  **matr[i] = (int\*)malloc(m\*sizeof(int));**  **// выделяется память под m элементов типа int**  **}**  **for(int i=0; i<n; i++){**  **free(matr[i]);**  **}**  **free(matr);** |

**for(int i=0; i<n; i++)**

**for(int j=0; j<m; j++){**

**matr[i][j] = i+j;**

**}**

**7.3. Пример: сортировка одномерного динамического массива**

**Реализация внутри функции main()**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<time.h>**

**int main(){**

**int n;**

**printf("Enter number of elements: ");**

**scanf("%i",&n);**

**//=========выделение памяти=========**

**int \*vector = new int [n];**

**//=========заполнение массива=========**

**srand(time(0));**

**//рандомизация – установка начального псевдослучайного числа**

**for(int i = 0; i < n; i++){**

**vector[i] = 1 + (rand() % 10);**

**//случайное число в диапазоне от 1 до 10**

**}**

**//=========проверка заполнения=========**

**for(int i = 0; i < n; i++){**

**printf("%i ",vector[i]);**

**}**

**printf("**\n**");**

**//=========сортировка=========**

**int i\_min, a;**

**for(int i = 0; i < n-1; i++){**

**i\_min = i; //наименьший - текущий**

**for(int j = i+1; j < n; j++){**

**if(vector[j] < vector[i\_min])**

**i\_min = j; //минимальный из еще неупорядоченных**

**}**

**//меняем местами элементы i и i\_min**

**a = vector[i];**

**vector[i] = vector[i\_min];**

**vector[i\_min] = a;**

**}**

**//=========вывод упорядоченного массива=========**

**for(int i = 0; i < n; i++){**

**printf("%i ",vector[i]);**

**}**

**printf("**\n**");**

**//=========освобождение памяти=========**

**delete [] vector;**

**return 0;**

**}**

**«Немного о функциях»**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<time.h>**

**//=========функция заполнения массива=========**

**int RandVector(int size, int\* massiv, int max\_rand){**

**srand(time(0));**

**//рандомизация – установка начального псевдослучайного числа**

**for(int i = 0; i < size; i++){**

**massiv[i] = 1 + (rand() % max\_rand);**

**//случайное число в диапазоне от 1 до 10**

**}**

**return 0;**

**}**

**//=========функция вывода массива на консоль=========**

**int PrintVector(int size, int\* massiv){**

**for(int i = 0; i < size; i++){**

**printf("%i ",massiv[i]);**

**}**

**printf("**\n**");**

**return 0;**

**}**

**//=========функция сортировки=========**

**int SortVector(int size, int\* massiv){**

**int i\_min, a;**

**for(int i = 0; i < size-1; i++){**

**i\_min = i; //наименьший - текущий**

**for(int j = i+1; j < size; j++){**

**if(massiv[j] < massiv[i\_min])**

**i\_min = j; //минимальный из еще неупорядоченных**

**}**

**//меняем местами элементы i и i\_min**

**a = massiv[i];**

**massiv[i] = massiv[i\_min];**

**massiv[i\_min] = a;**

**}**

**return 0;**

**}**

**int main(){**

**int n;**

**printf("Enter number of elements: ");**

**scanf("%i",&n);**

**//=========выделение памяти=========**

**int\* vector = new int [n];**

**//=========заполнение массива=========**

**RandVector(n,vector,10);**

**//=========проверка заполнения=========**

**PrintVector(n,vector);**

**//=========сортировка=========**

**SortVector(n,vector);**

**//=========вывод упорядоченного массива=========**

**PrintVector(n,vector);**

**//=========освобождение памяти=========**

**delete [] vector;**

**return 0;**

**}7.5. Пример: число четных элементов в двумерном динамическом массиве**

**int n = 5, m = 5;**

**int\*\* matr = new int\*[n];**

**// выделяется память под массив указателей на строки массива**

**for(int i=0; i<n; i++){**

**matr[i] = new int [m];**

**// выделяется память под m элементов типа int**

**}**

**// заполненеие**

**srand(time(0));**

**for(int i=0; i<n; i++)**

**for(int j=0; j<m; j++)**

**matr[i][j] = (rand() % 11) - 5;**

**// диапазон значений [-5,5]**

**// проверка заполнения**

**for(int i=0; i<n; i++){**

**for(int j=0; j<m; j++){**

**printf("%2i ", matr[i][j]);**

**}**

**printf("**\n**");**

**}**

**// подсчет числа четных элементов**

**int col\_odd = 0; // помните про вырожденные случаи!**

**for(int i=0; i<n; i++){**

**for(int j=0; j<m; j++){**

**if (matr[i][j] % 2 == 0)**

**col\_odd++;**

**}**

**}**

**printf("The sum of even elements: %i",col\_odd);**

**// освобождение памяти**

**for(int i=0; i<n; i++){**

**delete [] matr[i];**

}

**delete [] matr;**

**7.4. Строки**

***Нет стандартного типа данных «строка»!!!***

С++

(удобнее)

**#include<iostream>**

**#include<string>**

**using namespace std;**

**int main(){**

**string s = "В С++ есть клас string!";**

**cout << s;**

**return 0;**

**}**

С

(более эффективно)

**#include<stdio.h>**

**#include<string.h>**

**int main(){**

**char s1[80] = "hellow";**

**// выделено 80 байт**

**// первые элементы: 'h' 'e' 'l' 'l' 'o' 'w' '\0'**

**s1[0] = 'H';**

**char s2[] = "hellow";**

**// выделено 7 байт**

**s2[0] = 'H';**

**char\* s3 = "hellow";**

**// это указатель на строковую константу**

**// s3[0] = 'H' - ошибка времени выполнения**

**char \*s4 = new char[80];**

**s4[0] = 'H'; s4[1] = 'e'; s4[2] = 'l';**

**s4[3] = 'l'; s4[4] = 'o'; s4[5] = 'w';**

**s4[6] = '\0';**

**printf("%s\n%s\n%s\n%s\n",s1,s2,s3,s4);**

**return 0;**

**}**

**scanf("%s",s1);**

**scanf("%s",s4);**

**//без амперсанда!**

**//Имя массива (и статического, и динамического) –**

**//это указатель на его нулевой элемент!*Присвоить одну строку другой НЕЛЬЗЯ!***

**1.** В цикле посимвольно.

**2.** Функция **strcpy(s2,s1)**

(**string.h**, см. стр. 414-415).

**printf("Length of s4 = %i\n",strlen(s4));**

**printf("Enter a string: ");**

**gets(s1); //В чем отличие от scanf?**

1. **Построение новых типов**

**8.1. Переименование типов**

**typedef**

**typedef char MyString[80];**

**MyString str[10];**

**8.2. Перечисления**

**enum**

самостоятельно

**8.3. Структуры**

**struct**

**struct Sotrudnik{**

**char fio[80];**

**double zarplata;**

**};**

**Sotrudnik massiv\_sotr[100];**

**struct{**

**char fio[80];**

**double zarplata;**

**} massiv\_sotr[100];**

Тип полей структуры - …

**struct List{**

**int key;**

**List \*p; // указатель на эту же структуру**

**};**

**struct Sotrudnic{**

**char fio[80];**

**int zarplata;**

**} sotr1 = {"Ivanov", 10000}, sotr2, \*sotr = 0;**

**strcpy(sotr2.fio,"Petrov");**

**sotr2.zarplata = 100;**

**sotr2 = sotr1;**

**sotr = new Sotrudnic;**

**strcpy(sotr->fio,"Sidorov");**

**sotr->zarplata = 0;**

**8.4. Объединения**

**union** – все поля по одному адресу

**struct StecElement{**

**int element\_type;**

**union{**

**int i;**

**double d;**

**char ch;**

**};**

**};**

Ответственность за правильное использование – на программисте!!!

**struct{**

**int element\_type;**

**union{**

**int i;**

**double d;**

**char c;**

**};**

**} m[3];**

**m[0].element\_type = 1;**

**m[0].i = 10000;**

**m[1].element\_type = 2;**

**m[1].d = 1.1;**

**m[2].element\_type = 3;**

**m[2].c = 'q';**

**srand(time(0));**

**int i = rand() % 3;**

**switch (m[i].element\_type){**

**case 1: printf("%i**\n**",mas[i].i); break;**

**case 2: printf("%.1lf**\n**",mas[i].d); break;**

**case 3: printf("%c**\n**",mas[i].c); break;**

**default: printf("Error!**\n**");**

**}**