Указатели и ссылки

Практическое занятие

Объявление , определение, инициализация.

- Объявите указатели:
  На объект целого типа.
  На объект символьного типа.
  На объект вещественного типа двойной точности.
  Два указателя на целочисленный тип и три указателя на вещественный тип.
- Что будет объявлено в результате инструкций:
  typedef char * PCHAR;
  PCHAR p1, p2;
- Где компилятором будет созданы объекты:
  - **static int num = 0;**
  - **#include <stdio.h>**
    double val =0.5;
    main(){…}
  - **char * str =“D1J5”;**

Выполните инициализацию указателей:
Целочисленного типа,
Символьного типа,
Вещественного типа.

Напишите программу определяющую размер:
Целочисленного указателя,
Символьного указателя,
Вещественного указателя.
И вывести значения размера на экран.

Что записано в **n** на каждом шаге?

```
int x = 1;
int * p = &x;
int n = *(p++);
n = (*p)++;
n = ++(*p);
n = *(++p);
```

Арифметика указателей

Создать указатель *ptr на целочисленную переменную и присвоить значение через указатель 100. Скопировать значения указателя *ptr в указатель *ptr2. Присвоить значение 200 через указатель *ptr2. Вывести содержимое обоих указателей.

```
#include <iostream>
int main(int argc, char* argv[]){
   int* pt_var; // наш указатель
   int var = 515;      // значимая переменная типа int
   int second_var = 2;
// операция присваивания адреса переменной var указателю pt_var
   pt_var = &var;
   std::cout << "Косвенная операция сложения: " << * pt_var + second_var << std::endl;
   std::cout << "значение var = " << var << std::endl;
   return 0;
}
```

Еще задачка с указателями.

Что произойдет в результате компиляции и выполнении кода?

```
#include <stdio.h>
int main(){
    int a =5;
    int *b = &a;
    printf(“%d”, a**b);
     return 0;
}
```

Функции работы со строками в C (string.h)

Динамически выделяемая память под строку:

```
char *name = (char*)malloc(10);
scanf("%9s", name);
```

Или

```
char * gets(char *);
```

Для вывода :

```
char *str;
str = (char*)malloc(10);
…
 printf("%s", str);
int puts (char *str);
```

Основные функции стандартной библиотеки string.h

Функция	Описание
char ***strcat**(char *s1, char *s2)	присоединяет s2 к s1, возвращает s1
char ***strcpy**(char *s1, char *s2)	копирует строку s2 в строку s1, включая '\0', возвращает s1
int **strcmp**(char *s1, char *s2)	сравнивает s1 и s2, возвращает значение 0, если строки эквивалентны
int **strlen**(char *s)	возвращает количество символов в строке s
char ***strset**(char *s, char c)	заполняет строку s символами, код которых равен значению c, возвращает указатель на строку s

Пример работы со строками

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void main () {
   char *buffer = (char*)malloc(128);
   char *word = NULL;
   scanf("%127s", buffer);
   word = (char*) malloc(strlen(buffer)+1);
   strcpy(word, buffer);
   printf("%s", word);
   free(word);
   getch();
}
```

Копирует одну строку в другую, вместе с нулевым символом

Домашняя работа #6

- Выполнить сравнение 2-х строк, введенных с клавиатуры с игнорированием пробелов.

- *Для 2-х строк введенных с клавиатуры определить все возможные варианты вхождения второй строки в первую.