Тема: Классы и функции-члены

- Классы и структуры (**class** и **struct**)
- Создание типа (на примере **string**)

Классы и структуры

```
class TypeName {
    // неименованный раздел
  private:
    // закрытая часть
  public:
    // открытая часть
  protected:
    // защищенная часть
};
```

```
struct TypeName {
    // неименованный раздел
  private:
    // закрытая часть
  public:
    // открытая часть
  protected:
    // защищенная часть
};
```

Методы и свойства

```
class Point {

    public:
    // открытая часть – для методов
      void  setX( float );
      void  setY( float );
      float  getX( ) const;
      float  getY( ) const;
    private:
    // закрытая часть – для свойств
      float  x, y ;
};
```

```
void Point::setX( float val ) {
    x = val ;
}
void Point::setY( float val ) {
    y = val ;
}
float Point::getX( void ) {
    return x ;
}
float Point::getY( void ) {
    return y ;
}
```

Простой класс *String*

```
class String {

    public:

     void  set_to( char* );
     int  length( );
     int  read( );  // чтение из stdin
     void  print( );     // вывод в  stdout
    private:
     // детали реализации
};
```

Использование класса *String*

```
#include "mystring.h"
#include <stdio.h>
void main( int argc, char** argv ) {

    String  arg, input;
    if( argc == 2 ) {
      arg.set_to( argv[1] );
      printf( "Длина аргумента %d \n", arg.length() );
    }
    printf( "Введите строку: \n" );
    input.read( );
    printf( "Длина равна %d \n", input.length() );
}
```

Простая реализация

```
const int max_string_length = 128;

class String {
    public:
     void  set_to( char* );
     int  length( );
     int  read( );  // чтение из stdin
     void  print( );     // вывод в  stdout
    private:
     // объект типа String – это последовательность из не более чем
     // max_string_length символов, оканчивающаяся  символом NULL
     char text [ max_string_length+1 ];
};
```

Программирование методов

- Синтаксис

```
return_type   class_name::function_name ( parameter_list )
{
      // тело функции-члена класса
}
```

- Функция-член класса может ссылаться
  - на свои аргументы
  - на члены вызывающего объекта (не только открытые, но и закрытые свойства и методы)
    - с помощью ключевого слова this
    - по именам членов

Методы класса *String*

```
#include "mystring.h"
#include <string.h>  // для стандартных строковых функций
#include <stdio.h>
int String::length( void ) { return strlen( this->text ); }

void String::set_to( char* s ) {
    strncpy( text, s, max_string_length );
    (*this).text [max_string_length] = '\0';
 }
int String::read( void ) {
    if( fgets( text, max_string_length+1, stdin ) == 0 )
      return 0;     // конец файла
    if( text [length()-1] == '\n' ) text [length()-1] = '\0' ;
    return 1;
}
void String::print( void ) { fputs( text, stdout ); }
```

Расширение методов класса *String*

```
#include "mystring.h"
#include <stdio.h>
void main( int, char** ) {
    String  firstname, lastname, name;
    firstname.set_to ( "Bilbo");
    lastname.set_to ( "Baggins");
    name = firstname.concat ( &lastname );
    printf( "name is: " );
    name.print ( );
    firstname = name.substring ( 0, 5 );
    printf( "firstname is: " );
    firstname.print ( );
}
```

Добавление методов для *String*

```
const int max_string_length = 128;
class String {
    public:
      void  set_to( char* );
      int  length( );
      int  read( );  // чтение из stdin
      void  print( );     // вывод в  stdout
      String substring (int start, int len);
      // возвращает подстроку не изменяя оригинала
      String concat ( String* );
      // возвращает конкатенацию вызывающей строки   // и
    строки-аргумента, не изменяя их самих
    private:
      // объект типа String – это последовательность из не более чем
      // max_string_length символов, оканчивающаяся  символом NULL
      char text [ max_string_length+1 ];
};
```

Функция *String::substring*

```
#include "mystring.h"
#include <string.h>  // для стандартных строковых функций
#include <stdio.h>
#include <stdlib.h>  // для exit()
String String::substring ( int start, int len )  {
    if( start+len >= length() || start < 0 ) {
      fprintf( stderr, "illegal index (%d) for String \"%s\" \n",
      start, text );
      exit(1);
    }
    String sub;
    int i=0;
    while (i<len) { sub.text[i] = text[start+i] ;  i++;  }
    sub.text[i] = '\0 ';
    return sub;
}
```

Функция *String::concat*

```
#include "mystring.h"
#include <string.h>  // для стандартных строковых функций
#include <stdio.h>
#include <stdlib.h>  // для exit()

String String::concat ( String* s )  {
    if( length()+s->length() > max_string_length ) {
      fprintf( stderr, "string too large: \"%s %s\".\n",
      text, s->text );
      exit(1);
    }
    String both;
    strcpy( both.text, text );
    strcat( both.text, s->text );
    return both;
}
```

Упражнение

- Запрограммируйте простейший класс *Point*, реализующий понятие точки, как пары координат
- Координаты точки можно считать целочисленными
- Объявление класса разместите в файле ***point.h***, а определение – в файле ***point.cpp***
- Ограничьтесь тремя методами, обеспечивающими нижеследующий синтаксис

```
#include "point.h"
#include <stdio.h>
int main( int, char** )
{
   Point  p1, p2;
   p1.set_to ( 3, 5 ); p2.set_to ( 2, 7 );
   if ( p1.x() == 3 &&  p1.y() == 5  &&
        p2.x() == 2  &&   p2.y() == 7 )
   {
     printf( "test successful. \n" );   return 0;
   } else {
     printf( "test failed. \n" );      return 1;
   }
}
```

- Напишите пользовательскую функцию **print**, принимающую аргумент типа **Point** и печатающую значения **x** и **y**.

Упражнение

- Запрограммируйте рассмотренный в лекции класс *String*
- Добавьте метод is_the_same_as, который возвращает значение TRUE (ненулевое), если вызывающий объект равен аргументу и метод is_different_from, возвращающий TRUE (ненулевое), если вызывающий объект отличается от аргумента этой функции.
- Добавьте методы is_befor и is_after, которые определяют ASCII-упорядоченность строк
- Протестируйте следующий синтаксис:

```
#include “mystring.h"
#include <stdio.h>
void main(int, char *[]) {
    String h1, h2, w;
    h1.set_to("hello");
    h2.set_to("hello");
    w.set_to("world");

    printf("\n String::is_the_same_as ");
    if ( h1.is_the_same_as(h2) &&
       !h1.is_the_same_as(w) )
          printf("works.\n");
    else
          printf("doesn't work.\n");

    printf("\n String::is_different_from ");
         if (!h1.is_different_from(h2) &&
              h1.is_different_from(w))
          printf("works.\n");
    else
          printf("doesn't work.\n");

    printf("\n String::is_befor ");
    if ( !h1.is_befor(h2) &&
          h1.is_befor(w))
          printf("works.\n");
    else
          printf("doesn't work.\n");

    printf("\n String::is_after ");
    if ( !h1.is_after(h2) &&
         !h1.is_after(w))
          printf("works.\n");
    else
          printf("doesn't work.\n");
}
```