Объектно-ориентированное программирование с использованием языка C++

ШАГ компьютерная АКАДЕМИЯ

Урок №10

Объектно-ориентированное программирование с использованием языка C++

Содержание

Практический пример. Сохранение объектов структуры в файл

И, снова, здравствуйте!!! Для начала мы с вами сегодня закрепим прошлый материал и рассмотрим сохранение объекта структуры в файл. Приступим.

```
#include <iostream>
#include <string.h>
#include <stdio.h>
using namespace std;
//структура, хранящая
//информацию о человеке
struct Man{
    //Имя
    char str[255];
    //Возраст
    int age;
};

void main()
{
    //Создание объектов структуры
    Man A,B;
    //Запись в объект A
    //информации, полученной с клавиатуры
    cout<<"\nEnter name:\n";
    cin>>A.str;
    cout<<"\nEnter age:\n";
    cin>>A.age;
    //открытие файла на запись
    FILE*f=fopen("Test.txt","w+");
    if(!f) exit(0);
    //запись объекта A в файл
    fwrite(&A,sizeof(Man),1,f);
    c
    fclose(f);

    //открытие файла на чтение
    f=fopen("Test.txt","r+");
    if(!f) exit(0);
    //считывание содержимого файла
    //в объект B
    fread(&B,sizeof(Man),1,f);
    //открытие файла на запись
    fclose(f);

    //показ результата на экран
    cout<<"\nName - "<<B.str<<"\n\nAge - "<<B.age<<"\n\n";
}
```

Операции по работе с файлами

Мы с вами уже научились анализировать содержимое файла. Теперь пора научиться оперировать самими файлами. То есть, сейчас мы рассмотрим, такие операции, как переименование, перемещение, удаление и поиск файлов.

Набор функций для работы с файлами

stdio.h — переименование и удаление файлов

```
rename (char * oldname, char * newname)
```

Переименовывает файл.

oldname — путь и текущее имя файла.

newname — путь и новое имя файла.

Функция возвращает 0, если имя файла было успешно изменено, и ненулевое значение, если замена не произошла.

Примечание: Кстати!!! Если указать не только новое имя, но и новый путь — файл будет не только переименован, но и перенесён.

```
int remove(const char * filename)
```

Удаляет файл.

filename — путь и имя файла.

Функция возвращает 0, если имя файл был успешно удален, и ненулевое значение, если удаление не произошло.

Примечание: Помните!!! В момент удаления файл должен быть закрыт. Кроме того, не забывайте, что удаление необратимо.

io.h — поиск файлов

```
_findfirst(char * path, _finddata_t * fileinfo)
```

Находит файлы по указанному пути, соответсвующие маске.

Примечание: Маска — строка, критерий поиска, содержащая символы * (любая последовательность любых символов) и ? (любой один символ)

path — строка, в которой содержится комбинация пути и маски.

fileinfo — указатель на объект структуры, в который запишется информация о найденном файле. Структура содержит следующие поля:

1. **unsigned attrib** — содержит информацию об атрибутах файла.
   - _A_NORMAL — Обычный файл без запретов на чтение или запись.
   - _A_RDONLY — Файл только для чтения.
   - _A_HIDDEN — Скрытый файл.
   - _A_SYSTEM — Системный файл.
   - _A_SUBDIR — Директория.
   - _A_ARCH — Архивный файл.
2. **time_t time_create** — время/дата создания файла (равно –1 для FAT систем).

3. **time_t time_access** — время/дата последнего открытия файла (равно –1 для FAT систем).
4. **time_t time_write** — время/дата последнего редактирования файла.
5. **_fsize_t size** — размер файла.
6. **char name[260]** — имя файла.

Если всё пройдет успешно, информация о первом найденном файле запишется в объект структуры _finddata_t. При этом в оперативной памяти сформируется "список", обладающий внутренним указателем, который изначально будет установлен на первом найденном файле. В этом случае функция вернет уникальный номер, связанный с полученной группой файлов.

Если поиск завершится неудачей, функция вернет –1.

```
_findnext(long done, _finddata_t * fileinfo)
```

Функция осуществляет переход на следующий найденный файл в группе.

done — уникальный номер группы файлов в памяти.

fileinfo — указатель на объект структуры, в который запишется информация о следующем найденном файле.

Если достигнут конец списка файлов, функция вернет –1.

```
_findclose(long done)
```

Функция очищает память от группы найденных файлов.

done — уникальный номер группы файлов в памяти.

Пример программы на работу с файлами

```
//Здесь находятся функции переименования и удаления
#include <stdio.h>
//Здесь находятся функции для поиска файлов
#include <io.h>
#include <string.h>
#include <iostream>
using namespace std;

//Переименовать существующий файл
void RenameFile();

//Удалить существующий файл
void RemoveFile();

//Поиск файлов в каталоге
void Dir();

void main()
{
    //предлагаем выбрать пункт меню для выполнения
    cout << "Please, select preffer number...\n";

    //выбор пользователя
    char ch;
    do{
        //Переименовать
        cout << "\n1 - Rename\n";
        //Удалить
        cout << "2 - Remove\n";
        //Просмотреть некоторую папку(каталог)
        cout << "3 - View some folder\n";
        //Выход
        cout << "0 - Exit\n\n";
        cin >> ch;
```

```
        //анализируем и вызываем
        //соответствующую функцию
        switch(ch)
        {
            case '1':
                RenameFile();
            break;
            case '2':
                RemoveFile();
            break;
            case '3':
                Dir();
            break;
        }
    } while(ch != '0'); //Выход из программы
}

//Переименовать существующий файл
void RenameFile()
{
    char oldName[50], newName[50];

    //В одной переменной запомним существующее имя
    //(oldName),
    cout << "Enter old name:";
    cin >> oldName;

    //А в другой новое имя(newName)
    cout << "Enter new name:";
    cin >> newName;

    //Произведем переименование и проверку результата
    if (rename(oldName, newName) != 0)
        cout << "Error!\n Couldn't rename file.
            Check old and  new filename...\n\n";
    else
```



```
        cout << "Ok...\n\n";
}

//Удалить существующий файл
void RemoveFile()
{
    char Name[50];
    //Получаем имя и путь к удаляемому файлу
    cout << "Enter name:";
    cin >> Name;

    //Удаляем файл и проверяем результат
    if (remove(Name) != 0)
        cout << "Error!\n Couldn't remove file.
            Check filename...\n";
    else
        cout << "Ok...\n" ;
}

//Поиск файлов в каталоге
void Dir()
{
    //Запросим ПУТЬ (например, папка Temp на диске C,
    //запишется таким вот образом: c:\temp\)
    char path[70];
    cout << "\nEnter full path (for example, C:\\):";
    cin >> path;

    //Запросим маску файлов
    char mask[15];
    cout << "\nEnter mask (for example, *.* or *.txt):";
    cin >> mask;

    //Соединив две строки, мы получим результат
    //т.е. что хочет найти пользователь и где
    strcat(path, mask);
```

```
//Объявление указателя fileinfo
//на структуру _finddata_t
//и создание динамического объекта
//структуры _finddata_t
_finddata_t *fileinfo=new _finddata_t;

    //Начинаем поиск
    long done = _findfirst(path,fileinfo);

    //если done будет равняться -1,
    //то поиск вести бесмысленно
    int MayWeWork = done;

    //Счетчик, содержит информацию о количестве
    //найденых файлов.
    int count = 0;
    while (MayWeWork!=-1)
    {
        count++;
        //Распечатали имя найденного файла
        cout << fileinfo->name << "\n\n";
        //Пытаемся найти следующий файл из группы
        MayWeWork = _findnext(done, fileinfo);
    }
    //Вывод сообщения о количестве найденных файлов.
    cout << "\nInformation: was found " << count;
    cout << " file(s) in folder..." << path << "\n\n";

    //Очистка памяти
    _findclose(done);
    delete fileinfo;
}
```


Операции для работы с директориями

Кроме рассмотренных в предыдущем разделе урока файлах, существуют еще и папки, не так ли?! И, сейчас, мы с Вами займемся изучением приемов работы с папками (каталогами, директориями).

Набор функций

Библиотека direct.h

```
int _mkdir( const char *dirname )
```

Создает директорию по указанному пути.

dirname — Путь и имя для создаваемой директории.

```
int _rmdir( const char *dirname )
```

Удаляет директорию по указанному пути.

dirname — Путь и имя для удаляемой директории.

Обе функции возвращают –1 в случае ошибки.

Примечание: Кстати!!! Для переименования директории можно использовать функцию rename из библиотеки stdio.h.

Внимание!!! Удалить и переименовать можно только пустую директорию!!!

Пример на работу с директориями

```
#include <iostream>
#include <direct.h>
#include <stdio.h>

using namespace std;

//Переименовать существующую директорию
void RenameDirectory();

//Удалить существующую директорию
void RemoveDirectory();

//создать директорию
void CreateDirectory();

void main()
{
    //предлагаем выбрать пункт меню для выполнения
    cout << "Please, select preffer number...\n";

    //выбор пользователя
    char ch;
    do{
        //Переименовать
        cout << "\n1 - Rename\n";
        //Удалить
        cout << "2 - Remove\n";
        //Создать
        cout << "3 - Create\n";
        //Выход
        cout << "0 - Exit\n\n";
        cin >> ch;

        //анализируем и вызываем
        //соответствующую функцию
        switch(ch)
        {
            case '1':
                RenameDirectory();
                break;
            case '2':
                RemoveDirectory();
                break;
            case '3':
                CreateDirectory();
                break;
            }
    } while(ch != '0'); // Выход из программы
}

//Переименовать существующую директорию
void RenameDirectory()
{
    char oldName[50], newName[50];

    //В одной переменной запомним
    //существующее имя (oldName),
    cout << "Enter old name:";
    cin >> oldName;

    //А в другой новое имя(newName)
    cout << "Enter new name:";
    cin >> newName;

    //Произведем переименование и проверку результата
    if (rename(oldName, newName) != 0)
        cout << "Error!\n Couldn't rename directory.\n\n";
    else
        cout << "Ok...\n\n";
}

//Удалить существующую директорию
void RemoveDirectory()
{
```

```
    char Name[50];
    //Получаем имя и путь к удаляемой директории
    cout << "Enter name:";
    cin >> Name;

    //Удаляем директорию и проверяем результат
    if (_rmdir(Name) == -1)
        cout << "Error!\n Couldn't remove directory.\n";
    else
        cout << "Ok...\n" ;
}

//Создать директорию
void CreateDirectory()
{
    char Name[50];
    //Получаем имя и путь к создаваемой директории
    cout << "Enter name:";
    cin >> Name;

    //Создаем директорию и проверяем результат
    if (_mkdir(Name) == -1)
        cout << "Error!\n Couldn't create directory.\n";
    else
        cout << "Ok...\n" ;
}
```

Практический пример. Показ содержимого директории

Пример программы осуществляющей показ содержимого директории. Программа при запуске показывает содержимое текущей директории, а затем дает пользователю возможность выбора. Ввести можно будет следующие команды:

1. **cd Путь** — переход в другую директорию.
2. **cd ..** — показ содержимого родительского каталога и переход.
3. **cd** или **cd .** — показ содержимого текущего каталога.
4. **exit** — выход из программы.
5. **root** — переход в корневой каталог.

Примечание: Все вышеперечисленные команды регистронезависимы.

```
#include <iostream>
#include <windows.h>
#include <io.h>
#include <stdio.h>

using namespace std;

const int size=255;

//Функция, которая убирает лишние слеши и пробелы справа
void RemoveRSpacesAndRSlashes(char *str){
    int index=strlen(str)-1;
    while(str[index]=='\\'||str[index]==' '){
        index--;
    }
    strncpy(str,str,index);
    str[index+1]='\0';
}
//Функция для показа текущей директории
void ShowCurrentDir(char path[],char temp[]){
    CharToOem(path,temp);
    printf("%s>",temp);
}

//Функция перевода из кодировки
```

```
//Windows в кодировку DOS
//Для корректного отображения
//кирилицы
void RussianMessage(char path[]){
    CharToOem(path,path);
}

//Показ на экран содержимого папки
bool ShowDir(char path[]){
    //Показ содержимого текущей директории
    _finddata_t find;
    char pathfind[MAX_PATH];
    strcpy(pathfind,path);
    strcat(pathfind,"\\*.*");
    char info[MAX_PATH];

    //Начало Поиска
    int result=_findfirst(pathfind,&find);
    //Очистка экрана
    system("cls");
    int flag=result;
    if (flag==-1) {
        strcpy(info,"Такой Директории Нет");
        RussianMessage(info);
        printf("%s\n",info);
        return false;
    }

    while(flag!=-1){
        if(strcmp(find.name,".")&&strcmp(find.name,"..")){
        //Проверяем Директория или Нет
        find.attrib&_A_SUBDIR?strcpy(info," Каталог "):
                strcpy(info," Файл ");
            RussianMessage(info);
            RussianMessage(find.name);
            printf("%30s %10s\n",find.name,info);
        }
```



```
        //Продолжаем Поиск
        flag=_findnext(result,&find);
    }

    ShowCurrentDir(path,info);
    //Очищаем ресурсы, выделенные под поиск
    _findclose(result);
    return true;
}

void main(){
    //В данной переменной будет храниться
    //путь к Директории
    char path[MAX_PATH];
    //В данной переменной будет команда, введенная
    //пользователем
    char action[size];
    //Временная переменная
    char temp[MAX_PATH];
    //Получаем Путь к текущей Директории
    GetCurrentDirectory(sizeof(path),path);

    bool flag=true;

    //Показ содержимого текущей директории
    ShowDir(path);
    do{
        //Ввод команды пользователя
        cin.getline(action,size);
        //Убираем пробелы и слэши справа
        RemoveRSpacesAndRSlashes(action);
        //Переход в корневой каталог
        if(!strcmpi(action,"root")){
            path[2]='\0';
            ShowDir(path);
        }
        //Проверка на желание пользователя выйти
        else if(!strcmpi(action,"exit")){
```

```
            flag=false;
        }
        //Проверка на команду cd
        else if(!strnicmp(action,"cd",2)){
            //Показ содержимого текущей директории
            if((!strcmpi(action,"cd"))){
               //Показ Директории
               ShowDir(path);
            }
            //Команда cd была дана с параметрами
            else if(!strnicmp(action,"cd ",3)){
               //Находим индекс параметра
               int index=strspn(action+2," ");
               if(index){
                //Проверка на полный путь к Директории
                if(strchr(action+index+2,':')){
                //Попытка отобразить содержимое
                //Директории
                  if(ShowDir(action+index+2)){
                    strcpy(path,action+index+2);
                  }
                  else{
                    //Произошла Ошибка
                    ShowCurrentDir(path,temp);
                  }
                }
                //Поднимаемся в родительский каталог
                else if(!strcmp(action+index+2,"..")){
                  char *result=strrchr(path,'\\');
                  if(result){
                    int delta=result-path;
                    strncpy(temp,path,delta);
                    temp[delta]='\0';
                  }
                  else{
                    strcpy(temp,path);
                  }
```



```
                  if(ShowDir(temp)){
                    strcpy(path,temp);
                  }
                  else{
                    //Произошла Ошибка
                    ShowCurrentDir(path,temp);
                  }
                }
                //Показ Директории
                else if(!strcmp(action+index+2,".")){
                  ShowDir(path);
                }
                else if(!strcmp(action+index+2,"/")){
                  ShowDir(path);
                }
                else{
                  //Был Дан неполный путь
                  strcpy(temp,path);
                  strcat(temp,"\\");
                  strcat(temp,action+index+2);
                  //Попытка отобразить содержимое
                  //Директории
                  if(ShowDir(temp)){
                    strcpy(path,temp);
                  }
                  else{
                    //Произошла Ошибка
                    ShowCurrentDir(path,temp);
                  }
                }
            }
            else{
                //Показ Директории
                ShowDir(path);
                }
        }
        else{
```

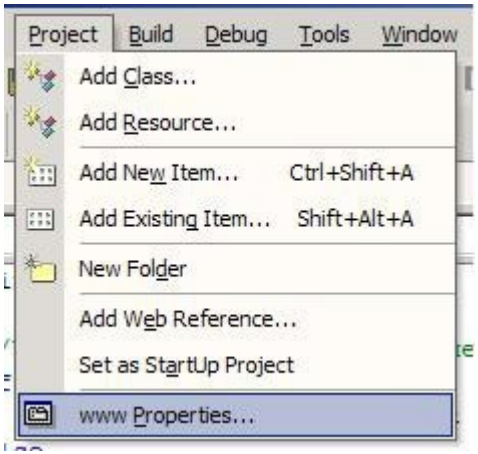
```
                //Показ Директории
                ShowDir(path);
            }
        }
        else{
            //Показ Директории
            ShowDir(path);
        }
    }while(flag);
}
```


Использование аргументов командной строки

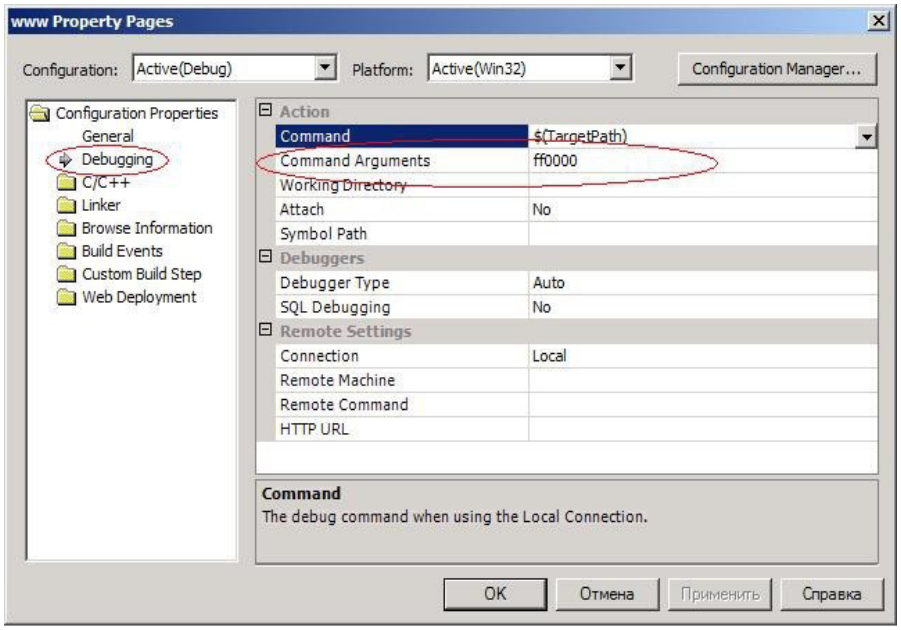
Существуют операционные системы, где имеется возможность передачи аргументов в функцию main из командной строки. Способов реализации данной передачи для наших с Вами приложений — три. Рассмотрим их.

Способ первый. Передача с помощью Visual Studio

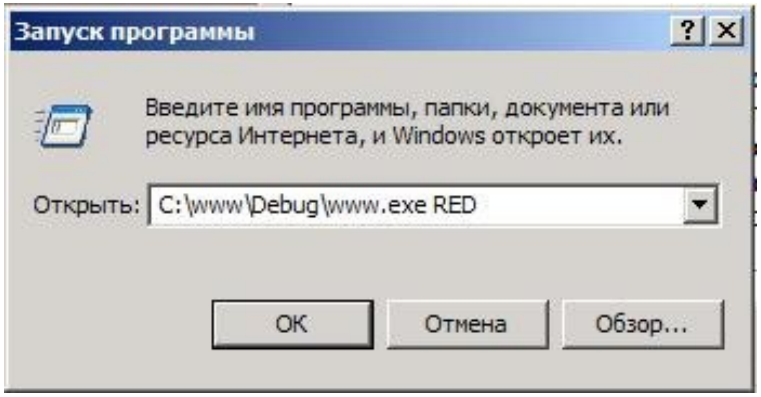
Этого необходимо зайти в меню Project и выбрать пункт **"имя проекта" Properties...**



Затем в открывшемся окне в секции Debugging в поле Command Arguments вписать аргументы командной строки.



Способ второй.
Передача с помощью команды "Выполнить" из меню "Пуск"



Способ третий.
Передача с помощью консоли (cmd.exe)

Обработка аргументов командной строки в функции main.

Для Обработки аргументов командной строки нужно в список параметров функции main включить два специальных параметра:

1. **int argc** — в данный параметр записывается значение, равное количеству аргументов командной строки.
2. **char *argv[]** — массив строк, в который помещаются непосредственно значения аргументов командной строки

Следует отметить, что в массиве argc присутствует всегда минимум один элемент — это путь к приложению.

Примечание: ВНИМАНИЕ!!! Названия параметров в main являются необязательными.

В следующем разделе урока мы разберем пример на использование аргументов командной строки.

Практический пример

Пример программы, которая создает HTML-документ и окрашивает его фон цветом переданным из командной строки в качестве параметра.

```
#include <iostream>
#include <string.h>
#include <stdio.h>

using namespace std;

void main(int argc, char * argv[])
{
    //Задаем по умолчанию черный цвет
    char str[7]="000000";
    //формируем начало HTML - документа
    char filehtml[256]="<html><head><title>
        New file!!!</title></head><body bgcolor = \'#";
    //Открываем файл на запись с созданием
    FILE*f=fopen("C:\\1.html","w+");
    //Если не получилось - останавливаемся
    if(!f) exit(0);
    //Если параметр цвета передан - используем его
    if(argc==2){
        strcpy(str,argv[1]);
    }
    //Присоединяем цвет к документу
    strcat(filehtml,str);
    //Присоединяем окончание к документу
    strcat(filehtml,"\'></body></html>");
    //Сохраняем в файл
    fputs(filehtml,f);
```



```
    //Закрываем Файл
    fclose(f);
    cout<<"\nOK.....\n";
}
```

Домашнее задание

1. Необходимо создать следующий набор программ:
   - Программа для копирования каталогов (копируются все вложенные папки и файлы).
   - Программа для перемещения каталогов (перемещаются все вложенные папки и файлы).
   - Программа для удаления каталогов (удаляются все вложенные папки и файлы).

   В том случае, если у удаляемого файла (каталога) установлен атрибут Read-Only, необходимо вывести следующее меню: 1. Удалять? 2. Пропустить? 3. Удалять для всех? 4. Отмена?
2. Для функций копирования и перемещения выполнить те же действия: 1. Перезаписать? 2. Пропустить? 3. Перезаписывать для всех? 4. Отмена?
   Параметры программы передаются через командную строку. Например,
   - copy.exe c:\A d:\B
   - move.exe c:\A d:\B
   - delete.exe c:\A