```cpp
// AllocNew1.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include <iostream>
#include <new>
using namespace std;

class myclass;
static void fun(myclass *pob, char *sstr[]);

class myclass
{
    char *name;
    int i;
public:
    myclass(char *nm, int ii);          //konstruktor sparametryzowany
    myclass();                          //przeciazenie konstruktora dla dynamicz. alok.
    ~myclass() { cout << "Niszczenie\n"; delete [] name;}
    void set_i(int ii) { i = ii; }
    void set_name(char *nm) { strcpy_s(name, _msize((void *)name), nm); }
    int get_i() { return i; }
    char *get_name() { return name; }
    void disp();
};

myclass::myclass(char *nm, int ii) : i(ii)
/*=========================================================================
To jest konstruktor sparametryzowany. Przeciez przy dynamicznej alokacji pamieci
ten konstruktor nie jest uzyteczny
=========================================================================*/
{
    cout << "Konstruktor sparametryzowany: Tworzenie\n";
    try
    {
        name = new char [512];
        memset((void *)name, 0, 512*sizeof(char));
        strcpy_s(name, sizeof(name), nm);
    }
    catch (bad_alloc xx)
    {
        cout << "memory allocation error" << endl;
        system("pause");
    }
}

myclass::myclass() : i(0)
/*=========================================================================
To jest konstruktor sparametryzowany. Przeciez przy dynamicznej alokacji pamieci
ten konstruktor nie jest uzyteczny
=========================================================================*/
{
    cout << "Konstruktor domyslny: Tworzenie\n";
    try
    {
        name = new char [512];
        memset((void *)name, 0, 512*sizeof(char));
    }
    catch (bad_alloc xx)
    {
        cout << "memory allocation error" << endl;
        system("pause");
    }
}

void myclass::disp()
{
    cout << name << endl;
}


int _tmain(int argc, _TCHAR* argv[])
{
    //deklarujemy wskaznik do myclass
```

```
    myclass *ptr_ob = NULL;

    cout << "Alokujemy pamiec operatorem new:\n";
    cout << "================================\n";
    //alokujemy tablice z 5 elementow typu obiektow myclass o nazwie ptr_ob
    try
    {
        ptr_ob = new myclass [5];
    }
    catch(bad_alloc aa)
    {
        cout << "Blad wydzielenia pamieci\n";
        system("pause");
        exit(1);
    }

    //przypisujemy dane
    char *str[] =
    {
        "aaaaa",
        "bbbbb",
        "ccccc",
        "ddddd",
        "eeeee"
    };

    fun(ptr_ob, str);

    //zwalniamy pamiec
    if(ptr_ob)
    {
        delete [] ptr_ob;
        ptr_ob = NULL;
    }

    cout << "\nAlokujemy pamiec operatorem new(nothrow):\n";
    cout << "=========================================\n";
    //alokujemy tablice z 5 elementow typu obiektow myclass o nazwie ptr_ob
    ptr_ob = new(nothrow) myclass [5];
    if(!ptr_ob)
    {
        cout << "Blad wydzielenia pamieci\n";
        system("pause");
        exit(1);
    }

    fun(ptr_ob, str);

    //zwalniamy pamiec
    if(ptr_ob)
    {
        delete [] ptr_ob;
        ptr_ob = NULL;
    }

    cout << "\nAlokujemy pamiec funkcja malloc:\n";
    cout << "================================\n";

    //Przy alokowaniu pamieci funkcja malloc konstruktor sie nie wywoluje
    //inicjowanie obiektu trzeba prowadzic recznie, uzywajac specjalne metody klasy
    //czas wydzielenia pamieci jest krotszy od alokacji przez operator new
    ptr_ob = (myclass *)malloc(5*sizeof(myclass));
    if(!ptr_ob)
    {
        cout << "Blad wydzielenia pamieci\n";
        system("pause");
        exit(1);
    }

    cout << "Uwaga! konstruktor sie nie wywoluje!\n";
    //to powoduje PAGE FAULT - pamiec dla tablicy name nie zaalokowana!

    fun(ptr_ob, str);
```

```
    //Przy zwolnieniu pamieci funkcja free destruktor sie
    //nie wywoluje. Jesli obiekt klasy zawiera dynamicznie alokowana
    //pamiec lub wskazniki do plikow, trzeba przed wywolaniem funkcji free
    //zwolnic ta pamiec, zamknac pliki
    if(ptr_ob)
    {
        free(ptr_ob);
        ptr_ob = NULL;
    }
    cout << "Uwaga! destruktor sie nie wywoluje!\n";

    system("pause");

    return 0;
}

void fun(myclass *pob, char *sstr[])
{
    int it;

    //wypelniamy tablice obiektow
    for(it=0; it<5; it++)
    {
        pob[it].set_i(it+1);
        pob[it].set_name(sstr[it]);
    }

    //wyswietlamy dane na monitorze
    cout << endl;
    cout << "--------------------------------" << endl;
    for(it=0; it<5; it++)
        pob[it].disp();
    cout << "--------------------------------\n" << endl;
}
```