ФГБОУ ВО “Чувашский государственный университет им. И. Н. Ульянова”
Факультет: ИВТ
Кафедра: Вычислительной техники
Предмет: Объектно-ориентированное программирование

Лабораторная работа №4

Указатели

Выполнил: студент группы ИВТ-41-20
Галкин Дмитрий
Проверил: доцент Обломов И.А.

Чебоксары 2021 г.

Теория

Указатели – особый вид переменных языка C++, содержащих в себе адреса ячеек памяти. Различают три вида указателей: указатель на объект какого-либо типа (стандартного или определенного пользователем), указатель на функцию и указатель на тип void.

- *Указатель на объект* – тип_указателя *имя_указателя = инициализация

Индивидуальное задание

1. Пользуясь результатами, полученными в первой лабораторной работе, объявить переменные стандартных типов. Объявить указатели на них, получить значения адресов объектов стандартного типа и значений этих объектов.

```
// Объявляем стандартные типы
char ch = '*';
int number_int = 1;
double number_double = 1.5;
bool isBool = 1;

// Объявляем указатели на стандартные типы и получаем их адреса
char *ptr_char = &ch;
int *ptr_int = &number_int;
double *ptr_double = &number_double;
bool *ptr_isBool = &isBool;

cout << "Declaring pointers (pointer, value) {" << endl;
cout << '\t' << &ptr_char << " = " << *ptr_char << endl;
cout << '\t' << &ptr_int << " = " << *ptr_int << endl;
cout << '\t' << &ptr_double << " = " << *ptr_double << endl;
cout << '\t' << &ptr_isBool << " = " << *ptr_isBool << " }" << endl;
```

2. Объявить переменные типа перечисление и указатели на объекты типа перечисления. Показать пример работы с этими переменными через указатели.

```
enum Days {
   MONDAY = 1
};
Days* my_days_ptr;
Days mun = MONDAY;
my_days_ptr = &mun;
std::cout << "Monday = " << *my_days_ptr << std::endl << std::endl;
```

3. Пользуясь структурой, объявленной в работе №1, объявить указатель на нее. Через указатель вывести на экран монитора значения всех полей структуры.

```
struct human {
   std::string name = "Dima";
} person;

struct human *student = &person;
std::cout << "Name = " << student->name << std::endl;
```

4. Исследовать, возможны ли преобразования переменных стандартных типов через указатели на них.

```
for (int i = 0; i < 10; i++) {
   std::cout << "Elem [ " << i << " ] = " << array_int[i] << std::endl;
}
// Indexing time: 27

for (int i = 0; i < 10; ++i) {
   std::cout << "Elem [ " << i << " ] = " << *(array_int + i) << std::endl;
}
// Pointer time: 18
```

5. Используя одномерный массив, описанный в лабораторной работе №3, организовать его обработку, пользуясь указателями.

```
int *arr_1 = new int[n]{1, -2, 5, 3, -7};
for (int i = 0; i < n; i++) {
   if (*(arr_1 + i) < 0) {
      countNegative++;
   }
   if (*(arr_1 + i) < 0 && flag == true) {
      index_2 = i;
      flag = false;
   }
   if (*(arr_1 + i) < 0 && flag == false && countNegative == 1) {
      index_1 = i;
      flag = true;
   }
}
```

6. Выполнить задание лабораторной работы №3 для многомерного массива, объявив его в динамической области памяти, используя операции new и delete.

```
int **arr = new int *[8];
arr[i] = new int[8];

void show (int **arr, const int n)
void sum(int **arr, int i)

for (int i = 0; i < 8; ++i) {
   delete [] arr[i];
}
delete [] arr;
```

7. Для пункта А) лабораторной работы №3 выполнить подпункт 1, передав одномерный массив как параметр функции. Выполнить обработку массива внутри функции согласно заданию.

```
typedef int(*PF)(int*, int);
void example(PF ptr_max, int *arr, int n) {
   std::cout << "Calling a func through a pointer (max arr element): " << ptr_max(arr, n);
}
```

8. Определить функцию, обрабатывающую массив по пункту А) и подпункту 2, передав его в качестве параметра. Объявить указатель на эту функцию и передать его некой другой функции, имеющей один из возможных параметров указатель на первую функцию.

```
typedef long int(*PS)(int*, int, int, int);
typedef long int(*PM)(PS, int*, int, int, int);

void example(PM multi_void, PS multi, int *arr, int n, int index_1, int index_2) {
   std::cout << "Multiplication of array elements: " << multi_void(multi, arr, n, index_1, index_2) <<
std::endl;
}

long int multi_example(PS ptr_multi, int *arr, int n, int index_1, int index_2) {
   return ptr_multi(arr, n, index_1, index_2);
}

example(multi_example, multiplication, arr_1, n, index_1, index_2);
```

9. Объявить объекты стандартных типов и указатели на них, а так же объекты тип структура, вывести значения этих объектов через указатели на них, и через указатель на тип void. Осуществить все необходимые преобразования.

```
void *ptr_void;

cout << "Declaring pointers (pointer, value) {" << endl;
ptr_void = ptr_char;
cout << '\t' << &ptr_char << " = " << *ptr_char << " or type void " << *(char*)ptr_void << endl;
ptr_void = ptr_int;
cout << '\t' << &ptr_int << " = " << *ptr_int << " or type void " << *(int*)ptr_void << endl;
ptr_void = ptr_double;
cout << '\t' << &ptr_double << " = " << *ptr_double << " or type void " << *(double*)ptr_void << endl;
ptr_void = ptr_isBool;
cout << '\t' << &ptr_isBool << " = " << *ptr_isBool << " or type void " << *(bool*)ptr_void << " }" <<
endl;

ptr_void = &person;

std::cout << "Name = " << (*(human*)ptr_void).name << std::endl;
std::cout << "Age = " << (*(human*)ptr_void).age << std::endl;
std::cout << "Sex = " << (((*(human*)ptr_void).sex == 1) ? "Man" : "Woman") << std::endl << std::endl;
```

Текст программы:

```
#include <iostream>
#include <string>
#include <random>
#include <ctime>
using namespace std;

typedef int(*PF)(int*, int);
typedef long int(*PS)(int*, int, int, int);
typedef long int(*PM)(PS, int*, int, int, int);
```

```
int max(int *arr, int n) {
    int max = -1000;
    for (int i = 0; i < n; i++) {
        if (*(arr + i) > max) {
            max = *(arr + i);
        }
    }

    return max;
}

void example(PF ptr_max, int *arr, int n) {
    std::cout << "Calling a func through a pointer (max arr element): " << ptr_max(arr, n);
}

void example(PM multi_void, PS multi, int *arr, int n, int index_1, int index_2) {
    std::cout << "Multiplication of array elements: " << multi_void(multi, arr, n, index_1, index_2) <<
std::endl;
}

long int multi_example(PS ptr_multi, int *arr, int n, int index_1, int index_2) {
    return ptr_multi(arr, n, index_1, index_2);
}



void sum(int **arr, int i) {
    long int sum = 0;
    for (int j = 0; j < 8; ++j) {
        sum += arr[i][j];
    }
    std::cout << i + 1 << " line = " << sum << std:: endl;
}

long int multiplication(int *arr, int n, int index_1, int index_2) {
    long int multiplication = 1;
    for (int i = index_1 + 1; i < index_2; i++) {
        multiplication *= *(arr + i);
    }

    return multiplication;
}

void transformation(int *arr_1, int *arr_2, int n) {
    int count = 0;
    for (int i = 1; i < n; i += 2) {
        *(arr_2 + count) = *(arr_1 + i);
        count++;
    }
    for (int i = 0; i < n; i += 2) {
        *(arr_2 + count) = *(arr_1 + i);
        count++;
    }
}
```

```
void show (int *arr, int n) {
    for (int i = 0; i < n; i++) {
        std::cout << *(arr + i) << ' ';
    }
    std::cout << std::endl;
}
void show (int **arr, const int n) {
    std::cout << "\t\tMatrix: " << std::endl;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; j++) {
            std::cout << arr[i][j] << ' ';
        }
        std::cout << std::endl;
    }
    std::cout << std::endl;
}


int main() {
    system("chcp 1251");
    std::cout << "/************************ Exercize 1 ************************/" << std::endl;

    // Declaring standard types
    char ch = '*';
    int number_int = 1;
    double number_double = 1.5;
    bool isBool = 1;

    // Declaring pointers
    char *ptr_char = &ch;
    int *ptr_int = &number_int;
    double *ptr_double = &number_double;
    bool *ptr_isBool = &isBool;

    cout << "Declaring standard types (name, value) {" << endl;
    cout << '\t' << "ch = " << ch << endl;
    cout << '\t' << "number_int = " << number_int << endl;
    cout << '\t' << "number_double = " << number_double << endl;
    cout << '\t' << "isBool = " << isBool << " }" << endl;

    cout << "Declaring pointers (pointer, value) {" << endl;
    cout << '\t' << &ptr_char << " = " << *ptr_char << endl;
    cout << '\t' << &ptr_int << " = " << *ptr_int << endl;
    cout << '\t' << &ptr_double << " = " << *ptr_double << endl;
    cout << '\t' << &ptr_isBool << " = " << *ptr_isBool << " }" << endl;

    cout << endl;
    std::cout << "/************************ Exercize 2 ************************/" << std::endl;

    enum Days {
        MONDAY = 1,
        TUESDAY = 2,
        WEDNESDAY = 3,
        FRIDAY = 4,
        SATURDAY = 5,
    };
```

```
Days* my_days_ptr;
Days mun = MONDAY;
my_days_ptr = &mun;
std::cout << "Monday = " << *my_days_ptr << std::endl << std::endl;


std::cout << "/********************** Exercize 3 **********************/" << std::endl;

struct human {
   // 1 - M, 0 - Ж
   std::string name = "Dima";
   int age = 19;
   int sex = 1;
} person;

struct human *student = &person;
std::cout << "Name = " << student->name << std::endl;
std::cout << "Age = " << student->age << std::endl;
std::cout << "Sex = " << ((student->sex == 1) ? "Man" : "Woman") << std::endl << std::endl;

std::cout << "/********************** Exercize 4 **********************/" << std::endl;

unsigned int start_time; // Начальное время
unsigned int end_time; // Конечное время

// Indexing
int array_int[10] = {23, 4, -378, 0, -64, 5, 11, 5, 9, 10};
start_time = clock();
for (int i = 0; i < 10; i++) {
   std::cout << "Elem [ " << i << " ] = " << array_int[i] << std::endl;
}
end_time = clock();
std::cout << "Indexing time: " << end_time - start_time << std::endl << std::endl;

// Pointering
start_time = clock();
for (int i = 0; i < 10; ++i) {
   std::cout << "Elem [ " << i << " ] = " << *(array_int + i) << std::endl;
}
end_time = clock();
std::cout << "Pointer time: " << end_time - start_time << std::endl;

std::cout << "/********************** Exercize 5 **********************/" << std::endl;

int n = 5;
int countNegative = 0;
bool flag = false;
int index_1 = 0, index_2 = 0;

int *arr_1 = new int[n]{1, -2, 5, 3, -7};
for (int i = 0; i < n; i++) {
   if (*(arr_1 + i) < 0) {
      countNegative++;
   }
   if (*(arr_1 + i) < 0 && flag == true) {
      index_2 = i;
```

```
            flag = false;
        }
        if (*(arr_1 + i) < 0 && flag == false && countNegative == 1) {
            index_1 = i;
            flag = true;
        }
    }

    // Paragraph A
    std::cout << "Maximum array element: " << max(arr_1, n) << std::endl;

    // Paragraph B
    if (countNegative >= 2) {
        std::cout << "Multiplication of array elements: " << multiplication(arr_1, n, index_1, index_2) <<
std::endl;
    } else {
        std::cout << "There are no negative elements in the array" << std::endl;
    }

    // Paragraph C
    int *arr_2 = new int[n];
    transformation(arr_1, arr_2, n);
    std::cout << "Array: ";
    show(arr_1, n);
    std::cout << "Transform array: ";
    show(arr_2, n);
    std::cout << std::endl;


    std::cout << "/************************ Exercize 6 ***********************/" << std::endl;

    // Random
    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_int_distribution<> dis(-2, 5);

    int **arr = new int *[8];
    countNegative = 0;
    for (int i = 0; i < 8; i++) {
        arr[i] = new int[8];
        for (int j = 0; j < 8; ++j) {
            countNegative++;
            arr[i][j] = dis(gen);
        }
    }

    show(arr, 8);

    // Paragraph 1
    std::cout << "k*k: ";
    for (int i = 0; i < 8; ++i) {
        std::cout << arr[i][i] << ' ';
    }
    std::cout << std::endl;

    // Paragraph 2
```

```
    if(countNegative == 0) {
        std::cout << "There are no negative elements in the array" << std:: endl;
    } else {
        for (int i = 0; i < 8; ++i) {
            for (int j = 0; j < 8; ++j) {
                if(arr[i][j] < 0) {
                    sum(arr, i);
                    break;
                }
            }
        }
    }

    std::cout << std::endl;

    std::cout << "/************************ Exercize 7 ************************/" << std::endl;

    // Paragraph A
    example(max, arr_1, n);

    std::cout << std::endl << std::endl;

    std::cout << "/************************ Exercize 8 ************************/" << std::endl;

    // Paragraph B
    if (countNegative >= 2) {
        example(multi_example, multiplication, arr_1, n, index_1, index_2);
    } else {
        std::cout << "There are no negative elements in the array" << std::endl;
    }

    std::cout << std::endl << std::endl;

    std::cout << "/************************ Exercize 9 ************************/" << std::endl;

    void *ptr_void;
    cout << "Declaring standard types (name, value) {" << endl;
    cout << '\t' << "ch = " << ch << endl;
    cout << '\t' << "number_int = " << number_int << endl;
    cout << '\t' << "number_double = " << number_double << endl;
    cout << '\t' << "isBool = " << isBool << " }" << endl;

    cout << "Declaring pointers (pointer, value) {" << endl;
    ptr_void = ptr_char;
    cout << '\t' << &ptr_char << " = " << *ptr_char << " or type void " << *(char*)ptr_void << endl;
    ptr_void = ptr_int;
    cout << '\t' << &ptr_int << " = " << *ptr_int << " or type void " << *(int*)ptr_void << endl;
    ptr_void = ptr_double;
    cout << '\t' << &ptr_double << " = " << *ptr_double << " or type void " << *(double*)ptr_void << endl;
    ptr_void = ptr_isBool;
    cout << '\t' << &ptr_isBool << " = " << *ptr_isBool << " or type void " << *(bool*)ptr_void << " }" <<
endl;

    ptr_void = &person;
    std::cout << "Name = " << (*(human*)ptr_void).name << std::endl;
    std::cout << "Age = " << (*(human*)ptr_void).age << std::endl;
```

```
    std::cout << "Sex = " << (((*(human*)ptr_void).sex == 1) ? "Man" : "Woman") << std::endl << std::endl;


    // Delete
    delete[] arr_1;
    delete[] arr_2;
    for (int i = 0; i < 8; ++i) {
        delete [] arr[i];
    }
    delete [] arr;

}
```

Вывод: Я практическим путем выяснил как использовать указатели на практики и к каким нежелательным последствиям могут привести неправильное использование указателей.