Занятие 6

Указатели. Управление динамической памятью с помощью `new` и `delete`.

Пример 1. *Указатели на основные типы данных.*
Для хранения любой переменной выделяется память. Все ячейки памяти пронумерованы, то есть имеют определенный адрес. Указатели (pointers) – переменные, значениями которых являются адреса памяти. Например, запись

```
int *p_a;
```

декларирует указатель с именем `p_a`. Значение этого указателя будет адрес памяти, куда записана переменная типа `int`. В примере выше выделяется память для переменной (указателя) `p_a`, но в выделенную память ничего не записывается.

Для получения адреса какой-либо переменной используется символ `&` перед именем переменной.

Для записи значения переменной по адресу, соответствующему значению указателя, используется символ `*` перед именем указателя. Такой же синтаксис используется для получения значения, записанного по адресу. Оператор `*`, пременённый к указателю, называется оператором разыменования (dereference operator).

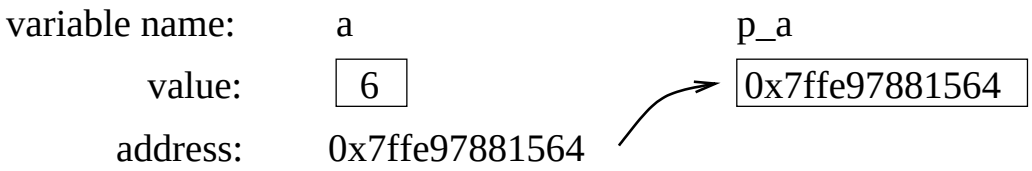

Рис. 1: Переменные и указатели.

```
#include <iostream>
int main(void){
  using namespace std;
  int a = 6; // объявление переменной и её инициализация
  int *p_a;  // объявление указателя на int (pointer to int)
  p_a = &a;  // присвоение указателю адреса int-переменной

  // Вывод на экран значения переменной двумя способами
  cout << "a = "    <<  a   << endl;
  cout << "*p_a = " << *p_a << endl;

  // Разыменование указателя.
  // Вывод на экран адреса переменной двумя способами
  cout << "&a = " << &a << endl;
  cout << "p_a= " << p_a << endl;

  // Изменение значения переменной через указатель
  *p_a = *p_a + 1;
  cout << "now a = " << a << endl;

  return 0;
}
```

Результат выполнения примера 1:

```
a = 6
*p_a = 6
&a = 0x7ffe97881564
p_a= 0x7ffe97881564
now a = 7
```

Пример 2. *Указатели и массивы.*

.

```
#include <iostream>
int main(void){
  using namespace std;
  const int N = 4;
  int a[N] = {10, 11, 12, 13}; // объявление и инициализация массива
  int *pa = a;                 // указатель на массив
  for (int i = 0; i < N; i++){
    cout << "a[" << i << "] = " << a[i] << endl;        // используя имя
    cout << "*(a+" << i << ") = " << *(a + i) << endl; // через указатель
    cout << "*pa=" << *pa << endl << endl;             // через указатель
    pa++;
  }
  return 0;
}
```

Пример 3. *Использование операции new для массивов.*
Если размер массива заранее (в момент написания программы) неизвестен, то для выделения необходимой памяти используется оператор `new`. Такое выделение памяти называется динамическим, так как осуществляется в процессе выполнения программы. Например, при выполнении строчки кода

```
double *a = new double [3];
```

выделяется память для хранения трех чисел типа `double` и адрес начала этого участка памяти присваивается указателю `a`. Вместо постоянного значения, 3 в данном примере, обычно используются значения целочисленных переменных.

```
#include <iostream>
int main(void) {
  using namespace std;

  double *a = new double[3]; // пространство для 3 значений double
  a[0] = 0.2;     // трактовать a как имя массива
  a[1] = 0.5;
  a[2] = 0.8;

  double *p = a;                          // инициализация еще одного указателя
  cout << "p[0]=" << p[0] << endl;
  p = p + 1;                              // увеличение указателя
  cout << "p[0]=" << p[0] << endl;
```

```
  p = p - 1;                          // возврат указателя в начало
  cout << "p[0]=" << p[0] << endl;
  delete[] a;                         // освобождение памяти

  return 0;
}
```

Пример 4. *Использование указателей на строки.*

```
#include <iostream>
#include <cstring>  // для strlen(), strcpy()

int main(void){
  using namespace std;
  char animal[20] = "bear";     // animal содержит bear
  const char *bird = "eagle";   // bird содержит адрес строки
  char *ps;                     // не инициализировано

  cout << animal << " and ";    // отображение bear
  cout << bird       << "\n";   // отображение eagle

  // cout << ps << "\n";        // может отобразить мусор, но может вызвать
                                // и аварийное завершение программы

  cout << "Enter a kind of animal (less than 20 symbols): ";
  cin >> animal; // нормально, если вводится меньше 20 символов

  // cin >> ps; // ошибка; ps указывает на невыделенное
                // пространство в памяти

  ps = animal;             // установка ps в указатель на строку
  cout << ps << "!\n";     // нормально; то же, что и применение animal
  cout << "Before using strcpy():\n";
  cout << animal << " at " << (int *)animal << endl;
  cout << ps     << " at " << (int *)ps     << endl;

  ps = new char[strlen(animal) + 1]; // динамическое выделение памяти
  strcpy(ps, animal);                // копирование строки в новое хранилище

  cout << "After using strcpy():\n";
  cout << animal << " at " << (int *)animal << endl;
  cout << ps     << " at " << (int *)ps     << endl;

  delete[] ps;  // надо освободить память
  return 0;
}
```

Результат выполнения примера 4:

```
bear and eagle
Enter a kind of animal (less than 20 symbols): lion
lion!
Before using strcpy():
lion at 0x7ffe4ec3c040
```

```
lion at 0x7ffe4ec3c040
After using strcpy():
lion at 0x7ffe4ec3c040
lion at 0x55faa6a64440
```

Пример 5. *Использование* `new` *со структурой.*

```cpp
#include <iostream>
struct candy_bar{ // определение структуры
  char name[20];
  float mass, price;
};

int main(void){
  using namespace std;
  candy_bar *psnack1 = new candy_bar; // выделение памяти для структуры

  cout << "Enter name of candy bar: ";// ввод имени, например, Snickers
  cin.get(psnack1->name, 20);         // первый метод для доступа к членам

  cout << "Enter mass (grams): ";     // ввод массы, например, 100 грамм
  cin >> (*psnack1).mass;             // второй метод для доступа к членам

  cout << "Enter price (rubles): ";   // ввод цены
  cin >> psnack1->price;

  cout << "Name: "  << (*psnack1).name << endl;              // второй метод
  cout << "Mass: "  << psnack1->mass   << " grams\n";        // первый метод
  cout << "Price: " << psnack1->price  << " rubles" << endl; // первый метод

  delete psnack1;   // освобождение памяти
  return 0;
}
```

Задания

Задание 6.1. Напишите программу, которая запрашивает у пользователя, сколько целых чисел он хочет ввести, создает динамически массив для их хранения и заполняет его, запрашивая значения у пользователя. После этого программа выводит значения на экран и освобождает память.

Задание 6.2. Напишите программу, которая собирает информацию о книгах (см. задание 3.3). Сначала программа запрашивает, сколько книг требуется ввести, и выделяет память для соответствующего массива структур. Затем программа запрашивает последовательно информацию о каждой книге. После этого вся информация выводится на экран и память освобождается.