**Контрольні запитання**

1. **Покажчик (pointer) - спеціальна змінна в якій може зберігатися адреса оперативної пам'яті, тобто адреса початкового байта будь-якої іншої змінної або об'єкта.**

2. Для **объявления** переменной-указателя **перед** именем переменной необходимо поставить символ **\*** . Имена указателей рекомендуется начинать с **p** (pAge, pNumber).

3. Для получения адреса переменных в С++ используется операция адрес - **&.**

4. Для помещения адреса переменной (объекта) в указатель необходимо применять операцию адрес (**&**). Выполнить это можно путем инициализации указателя или оператором присваивания.

5. Указатель, значение которого **0** (**NULL**) называется **нуль*-*указателем** (**null pointer**). Все указатели при их создании лучше **инициализировать** каким-то значением – **адресом** переменной. Если вы этого адреса не знаете, то присвойте указателю значение 0.

6. Присвоить 0.

8. Неинициализированные указатели называют **висящими** указателями (**wild pointers**) – они очень опасны.  
9. При операции **разыменования** - символ **\*** перед именем говорит о том, что нам нужен доступ к значению, находящемуся в памяти по адресу, хранящемуся в указателе, а не сам адрес. Например: **\*pAge=5**; - присваивание **5** значению по адресу, который хранится в **pAge**. Этот же символ \* используется и в операции умножения. Компилятор **С++** распознает нужную операцию по контексту.

10.

11. Указатели используют в **следующих** случаях :

* работа с данными в свободной памяти;
* доступ к данным-членам и функциям объектов в свободной памяти;
* передача переменных функциям по ссылке.

12. Для выделения области свободной памяти используется оператор **new**, за которым записывается тип объекта, под который вы хотите выделить память. Значением, которое возвращает **new**, является адрес области свободной памяти. Это значение должно быть присвоено указателю.

13. Если **new** не может выделить память в свободной зоне, то возвращается нуль-указатель (**NULL**). При каждом запросе программы новой свободной памяти вы должны проверять ваш указатель на **NULL**.

14. Для **освобождения** участка свободной памяти используется оператор **delete**. Этот оператор **освобождает** память, адрес которой хранился в указателе

15.

16. **Утечка памяти (англ. memory leak)** — это неконтролируемое уменьшение свободной оперативной или виртуальной памяти компьютера. Причиной утечек становятся ошибки в программном коде.

/\*1\*/ char \*pointer = NULL;

/\*2\*/ for( int i = 0; i < 10; i++ ) {

/\*3\*/ pointer = new char[100];

/\*4\*/ }

/\*5\*/ delete [] pointer;

В этом примере на 3-й строке создается объект в динамической памяти. Код на 3-й строке выполняется 10 раз, причём каждый следующий раз адрес нового объекта перезаписывает значение, хранящееся в указателе pointer. На 5-й строке выполняется удаление объекта, созданного на последней итерации цикла. Однако первые 9 объектов остаются в динамической памяти, и одновременно в программе не остаётся переменных, которые бы хранили адреса этих объектов. Т.е. в 5-й строке невозможно ни получить доступ к первым 9 объектам, ни удалить их.

17. Висячие указатели появляются при перемещении или удалении объектов, если при этом значение указателя не изменено на нулевое. Даже если система перераспределит освобожденную память под другие данные, указатель все равно будет ссылаться на прежние ячейки. Это может привести к непредсказуемому поведению программы.