Санкт-Петербургский государственный университет

Вопросы и ответы к зачёту 9 ноября 2023 г. № 1

№ 2 будет в декабре

Санкт-Петербургский государственный университет

Вопросы

1. Что такое язык C?
2. Что такое язык C++?
3. Какие знаете классификации языков программирования? Приведите примеры и языков.
4. Что такое низкоуровневый язык?
5. Что такое высокоуровневый язык?
6. Что такое компиляция программы?
7. Что такое интерпретация программы?
8. Что такое машинный код?
9. Что такое байт-код?
10. Какие сортировки вы знаете?

Санкт-Петербургский государственный университет

Вопросы

11. Что такое сложность алгоритма и от чего она зависит?
12. Реализуйте любую сортировку. Какая у неё сложность?
13. Зачем в C++ (C) код модулей разделяют на *.hpp (*.h) и *.cpp (*.c)?
14. Как проходит процесс компиляции cpp файлов в бинарный файл?
15. Что такое препроцессор и как он работает?
16. Что такое директивы (команды) препроцессора и какие знаете?
17. Как работает директива include?
18. Что такое макрос, какие макросы бывают и как работают?
19. Как работает директива define?
20. Как защитить хедер от повторного включения?

Санкт-Петербургский государственный университет

Вопросы

21. Что линкует линкер?
22. Что такое императивное программирование?
23. Что такое процедурное программирование?
24. Что такое структурное программирование?
25. Что такое ООП?
26. Как можно передавать параметры в функцию? Кратко поясните
27. Что такое передача параметров в функцию по ссылке?
28. Что такое передача параметров в функцию по указателю?
29. Что такое указатель?
30. Что такое ссылка?

Санкт-Петербургский государственный университет

Вопросы

31. Что такое рефакторинг?
32. Что такое область видимости и какие бывают области видимости?
33. Что такое пространство имён и что такое std?
34. Как const влияет на переменные? Объясните что значит «const char* const comment»?
35. Как static влияет на глобальные и локальные переменные?
36. Какие основные типы данных есть в C++ и сколько они занимают памяти?
37. Чем отличается префиксный инкремент (декремент) от постфиксного?
38. Что такое разрядность машины?
39. Сколько места в памяти отводится под хранение указателя?
40. Что означает тип указателя?

Санкт-Петербургский государственный университет

Вопросы

41. Где хранятся переменные программы?
42. Что такое стек (память)?
43. Что такое куча (память)?
44. Что такое массив и чем он отличается от вектора?
45. Чем отличается массив от указателя на массив?
46. Как найти размер массива?
47. Чем отличаются delete и delete[]?
48. Что будет, если забыть вызвать delete? Когда освободится та память?
49. Как массив передаётся в функцию?
50. Как происходит взятие элемента массива по индексу?

Санкт-Петербургский государственный университет

Что произойдёт в коде?

51

```cpp
#include <iostream>

#define MAX 10

int main( ) {
    int n;
    n = ++MAX;
    std::cout << n;
}
```

Ошибка компиляции: Оператор ++ можно применять только к изменяемым переменным

Нельзя изменить константу MAX

52

```cpp
#include <iostream>

int main( ) {
    const int a = 5;
    std::cout << a++;
}
```

Ошибка компиляции: Оператор ++ можно применять только к изменяемым переменным

Нельзя изменить константу a

Санкт-Петербургский государственный университет

Что произойдёт в коде?

53

```cpp
#include <iostream>

int main() {
    int i = 10;
    if (i = 20) {
        std::cout << i;
    }
}
```

На консоль выведется 20

В условии if происходит не сравнение на равенство «==», а операция присвоения «=». При этом, в if будет возвращено значение i. И т.к. i будет не нулевым, то оно будет преобразовано в true

54

```cpp
#include <iostream>

int main() {
    int a = 5, b = 6, c;
    c = a > b ? a : b;
    std::cout << c;
}
```

На консоль выведется 6

Тернарный оператор ?

Санкт-Петербургский государственный университет

Что произойдёт в коде?

55

```cpp
#include <iostream>

int main() {
    int x = 4;
    if (x == 4) {
        if (x == 4) {
            break;
            std::cout << "Buy-buy!";
        }
    }
    std::cout << "Buy!";
}
```

Ошибка компиляции: оператор break можно использовать только внутри циклов или выражения для выбора вариантов (switch)

56

```cpp
#include <iostream>

int main() {
    int i = 1;
    do {
        std::cout << i;
        i++;
        if (i < 3) continue;
    } while (false);
}
```

На консоль выведется 1

Оператор continue отправляет выполнение не в начало следующей итерации, а в конец текущей итерации. Т.е. после выполнения 8-ой строчки выполнение перейдет на 9-ю строчку, где происходит проверка условия

Санкт-Петербургский государственный университет

Что произойдёт в коде?

57

```cpp
#include <iostream>

int main() {
    char x[3] = "12345";
    std::cout << x;
}
```

Ошибка компиляции: нельзя использовать const char[6] для инициализации char[3]

x — это массив 3 char элементов. Т.е. под x выделено последовательно в памяти 3 байта. А ему пытаются присвоить 6 char элементов. 6 потому, что строка, заключённая в двойные кавычки, всегда содержит элемент окончания строки '\0'

58

```cpp
#include <iostream>

int main() {
    char str[5] = "ABC";
    std::cout << str[3] << std::endl;
    std::cout << str << std::endl;
}
```

На консоль выведется:

'\0' (пустой символ)
ABC

Строка, заключённая в двойные кавычки, всегда содержит элемент окончания строки '\0'. Поэтому str[3] будет '\0'. И все остальные неинициализированные элементы будут тоже '\0'. А вот начиная с str[5] и далее ячейки памяти будут содержать мусорные значения

Санкт-Петербургский государственный университет

Что произойдёт в коде?

59

```cpp
#include <iostream>

int arr_global[100];

int main() {
    int arr_local[100];
    static int arr_static_local[100];

    std::cout << arr_local[99] << std::endl;
    std::cout << arr_static_local[99] << std::endl;
    std::cout << arr_global[99] << std::endl;
}
```

На консоль выведется:

мусорное значение
0
0

Локальные переменные не имеют инициализации по умолчанию, а глобальные или статические локальные переменные инициализируются 0

60

```cpp
#include <iostream>

int main() {
    int arr[] = { 10, 20, 30 };
    std::cout << *arr + 1;
}
```

На консоль выведется: 11

Санкт-Петербургский государственный университет

Что произойдёт в коде?

61

```cpp
#include <iostream>

int main() {
    std::cout << 2["HELLO"];
}
```

На консоль выведется: L

Т. к. в C и C++ массив отображает последовательность ячеек памяти, то взятие индекса — это сдвиг указателя массив. И получаем равенство: arr[i] = *(arr + i) = *(i + arr) = i[arr]

62

```cpp
#include <iostream>

int main() {
    int arr[] = { 10, 20, 30 };
    std::cout << -2[arr];
}
```

На консоль выведется: -30 (а не ошибка!)

«-» перед 2 — это унарный оператор минус, который имеет приоритет меньше, чем оператор []. Поэтому вначале будет выполнено 2[arr], а потом «-» минус

Т. к. в C и C++ массив отображает последовательность ячеек памяти, то взятие индекса — это сдвиг указателя массив. И получаем равенство: arr[i] = *(arr + i) = *(i + arr) = i[arr].

Санкт-Петербургский государственный университет

Ответы

1 Что такое язык Си?

Си — это высокоуровневый, процедурный, компилируемый, статически типизированный язык программирования, разработанный в 1969-1973 годах Деннисом Ритчи.

Замечание: одни считают Си высокоуровневым, а другие низкоуровневым, т. к. немного расширяют понятие низкоуровневого языка, а именно: к низкоуровневым причисляют также все языки, которые позволяют писать достаточно оптимальный код для процессора. Аналогичная проблема и с Фортраном. И тот и другой ответ с данным пояснением будет правильным.

2 Что такое язык C++?

C++ — это высокоуровневый, мультипарадигмальный, компилируемый, статически типизированный язык программирования, разработанный в 1983 году Бьёрном Страуструпом.

3 Какие знаете классификации языков программирования? Приведите примеры и языков.

Языки программирования можно разделить на низкоуровневые (язык Ассемблера) и высокоуровневые (C, C++, Java, Python, JavaScrypt), компилируемые (C, C++) и интерпретируемые (Python, JavaScrypt). Язык Java в последней классификации относится к языкам «компилируемым в байт код» и интерпретируемым этого байт-кода в машинный код во время выполнения.

4 Что такое низкоуровневый язык?

Низкоуровневый язык программирования — это язык близкий к программированию на машинных кодах. Для обозначения машинных кодов применяются мнемоники. Низкоуровневые языки привязаны к конкретной реализации вычислительной системы.

Пример — язык Ассемблера

Санкт-Петербургский государственный университет

Ответы

5 Что такое высокоуровневый язык?

Высокоуровневый язык — это язык, который имеет существенную абстракцию от машинных команд и имеет человекоподобный вид, позволяя тем программисту намного быстрее и удобнее писать код.

6 Что такое компиляция программы?

Компиляция — это сборка программы, которая состоит из перевода программы в машинный код и компоновку итоговой исполняемой машинной программы с учётом использованных в исходном коде библиотек.

7 Что такое интерпретация программы?

Интерпретация — это метод выполнения программы, при котором каждый отдельно взятый оператор транслируется в машинный код и сразу выполняется, после чего осуществляется переход . Из-за этого программы написанные на интерпретируемых языках (Java, C#, Python, JavaScrypt) работают медленнее, чем на компилируемых.

8 Что такое машинный код?

Машинный код — система команд конкретной вычислительной машины, которая выполняется непосредственно процессором.

9 Что такое байт-код?

Байт-код — промежуточное представление, в которое может быть переведена компьютерная программа. Байт-код похож на машинный код, но предназначен для исполнения не реальным процессором, а виртуальной машиной. В качестве виртуальной машины выступает интерпретатор соответствующего языка программирования. В байт-код переводятся программы, написанные на интерпретируемых языках. У разных интерпретируемых языков разный байт-код.

Санкт-Петербургский государственный университет

Ответы

10 Какие сортировки вы знаете?

Сортировку пузырьком, вставками, выбором, сортировку Шелла, сортировку слиянием, сортировку кучей и быструю сортировку.

11 Что такое сложность алгоритма и от чего она зависит?

Сложность алгоритма оценивают по времени выполнения и/или по используемой памяти. В обоих случаях сложность зависит от размера входных данных. Для оценки сложности алгоритма по времени используют O-нотацию (О большую), которая показывает как время работы растёт в зависимости от объёма входных данных. Например, $O(n^2)$ — количество операций зависит от размера массива как $n \cdot n$, $O(\log(n))$, $O(n \cdot \log(n))$.

12 Реализуйте любую сортировку. Какая у неё сложность?

Сортировка пузырьком (по возрастанию). Её сложность $O(n^2)$. Пусть есть массив *a* длиной *n*, тогда:

```
1 for (int i = 0; i < n - 1; i++ ) {
2     for (int j = 0; j < n - i - 1; j++) {
3         if (a[j] > a[j + 1]) {
4             int x = a[j];
5             a[j] = a[j + 1];
6             a[j + 1] = x;
7         }
8     }
9 }
```

Санкт-Петербургский государственный университет

Ответы

13 Зачем в C++ (C) код модулей разделяют на *.hpp (*.h) и *.cpp (*.c)?

Файлы с расширением *.hpp (*.h) – файлы заголовков – используются для объявления (declaration) функций, классов и других сущностей. А файлы *.cpp (*.c) используются для определения / реализации (definition). Нельзя использовать функцию без и до её объявления в файле. Это вызывет ошибку на этапе компиляции (при анализе кода перед переводом его на машинный язык). Если функция будет объявлена, но не будет определена (.cpp файл), то это вызывет ошибку на этапе линковки.

Функцию можно объвить 3 способами:

1. В каждом файле, где используется

```cpp
int func();

int main() {
  std::cout << func();
}
```

2. В файле определения (*.cpp / *.c)

```cpp
#include "func.cpp"

int main() {
  std::cout << func();
}
```

3. В файле объявления (*.hpp / *.h)

```cpp
#include "func.hpp"

int main() {
  std::cout << func();
}
```

В 1-ом случае, если поменяется что-то в сигнатуре функции, то это изменение придётся вносить во все файлы, где функция используется.

Во 2-ом случае при изменении сигнатуры функции изменение в объявлении надо сделать только один раз в cpp, но директива #include вставит много ненужного текста.

В 3-ем случае оба момента из 1-го и 2-го вариантов будут оптимальными для разработчика и компилятора. Поэтому и были созданы заголовочные файлы *.hpp (*.h) – только для объявления сущностей.

Санкт-Петербургский государственный университет

Ответы

14 Как проходит процесс компиляции cpp файлов в бинарный файл?

Процесс компиляции cpp файлов проходит в 3 этапа:

1. Препроцессинг
2. Трансляция
3. Линковка

Во время препроцессинга в исходном файле происходит удаление комментариев, замена макросов на соответствующие тексты, включение необходимого текста библиотек и своих хедеров. При этом, нового файла не создается, а происходит преобразование существующего cpp.

Во время трансляции происходит лексический анализ кода и затем перевод программы с языка C++ на машинный язык для каждого из cpp файлов. В результате для каждого cpp файла создаётся свой obj файл. В obj файлах отсутствует связь с другими модулями, реализация библиотечных функций, код запуска всей программы.

Во время линковки связываются все obj-ые файлы в единый исполняемый файл, добавляет код запуска для конкретной ОС, и добавляет реализации необходимых библиотечных функций.

15 Что такое препроцессор и как он работает?

Препроцессор — это программа, принимающая данные на входе и выдающая данные для другой программы. В C и C++ препроцессор — это часть компилятора, которая преобразует исходных код перед его трансляцией в машинный код. Препроцессор в исходном файле производит удаление комментариев, замену макросов на соответствующие тексты, включение необходимого текста библиотек и своих хедеров. При этом, нового файла не создается, а происходит преобразование существующего cpp.

Санкт-Петербургский государственный университет

Ответы

16 Что такое директивы (команды) препроцессора и какие знаете?

Директивы препроцессора — это команды, которые используются для упрощения изменения текста исходных программ.

#define, #include, #ifndef, #undef, #pragma и т. д.

17 Как работает директива include?

Указывает препроцессору включить содержимое указанного файла в точку, где появляется директива. Есть два вида: #include “my_file.hpp” — для включения своих файлов, #include <out_lib> — для включения внешних библиотек.

18 Что такое макрос, какие макросы бывают и как работают?

Макрос — это идентификатор, который представляет константу или выражение. Он создаётся директивой препроцессора #define. Макросы обрабатывает препроцессор, который вместо имени макроса в коде подставляет тело макроса.

19 Как работает директива define?

Директива define создает макрос, который может использоваться для представления константы или выражения. Препроцессор подставляет тело макроса в место каждого его обнаружения в коде.

С помощью макросов можно создавать не только изящный код, но и плодить не менее изящные баги.

Санкт-Петербургский государственный университет

Ответы

20 Как защитить хедер от повторного включения?

Защитить .hpp файл от повтороного включения можно либо директивой препроцессора #pragma once, либо конструкцией #ifndef, #define, #endif. Директива #pragma once может быть не реализована в некоторых компиляторах. Но если она реализована, то лучше использовать её, т. к. в таком случае глобальное пространство имён будет меньше засоряться.

21 Что именно линкует линкер?

Линкер связывает все obj-ые файлы в единый исполняемый файл, добавляет код запуска для конкретной ОС, и реализации необходимых библиотечных функций.

22 Что такое императивное программирование?

Императивное программирование — это парадигма (стиль, подход к написанию программ), когда последовательно выполняется последовательность команд, и результаты предыдущих команд могу быть прочитаны последующими.

*Любой язык программирования в той или иной степени является императивным. Самое чистое императивное программирование — это написание программы на машинном языке.

23 Что такое процедурное программирование?

Процедурное программирование — это парадигма, позволяющая писать процедуры или функции.

Процедура может иметь входные аргументы, но не может иметь выходных.

А функция может иметь и выходные аргументы.

Санкт-Петербургский государственный университет

Ответы

24 Что такое структурное программирование?

Структурное программирование — это парадигма, которая позволяет писать программу, как набор блоков. Программа представлена в виде иерархической структуры блоков. Базовыми блоками здесь являются последовательность, ветвление, циклы, подпрограммы в виде процедур и функций и блоки кода.

Оператор goto – это враг структрного программирования, т. к. структурное программирование — это чёткая понятная логика построения программы.

Первый язык, который предоставил возможность писать программы в парадигме структурного программирования стал C.

25 Что такое ООП?

ООП — это парадигма, основанная на представлении программы в виде совокупности взаимодействующих объектов, каждый из которых является экземпляром определённого класса. ООП позволяет писать программы с более сложной структурой, чем структурное программировани. Оно вводит понятия классов и объектов, наследования, инкапсуляции и полиморфизма.

Самым известным и одним из самых первых языков, которые позволяют писать программы в парадигме ООП, является C++.

26 Как можно передавать параметры в функцию? Кратко поясните

Параметры в функцию могут быть переданы по значению, по ссылке, по указателю.

При передаче параметров по значению функция получает копию значения переменных.

При передаче параметров по ссылке передается ссылка на объект, через которую мы можем манипулировать самим объектов, а не просто его значением.

Указатели передаются в функцию по значению, то есть функция получает копию указателя. В то же время копия указателя будет в качестве значения иметь тот же адрес, что оригинальный указатель.

Санкт-Петербургский государственный университет

Ответы

27 Что такое передача параметров в функцию по ссылке?

При передаче параметров по ссылке передается ссылка на объект, через которую мы можем манипулировать самим объектов, а не просто его значением: void func(int &a)

Передача по ссылке позволяет возвратить из функции сразу несколько значений. Также передача параметров по ссылке является более эффективной при передаче очень больших объектов. Поскольку в этом случае не происходит копирования значений, а функция использует сам объект, а не его значение.

28 Что такое передача параметров в функцию по указателю?

Указатели передаются в функцию по значению, то есть функция получает копию указателя. В то же время копия указателя будет в качестве значения иметь тот же адрес, что оригинальный указатель. Поэтому используя в качестве параметров указатели, мы можем получить доступ к значению аргумента и изменить его.

void func(int *a)

Для изменения значения параметра применяется операция разыменования с последующим инкрементом: *arr

Поскольку теперь функция в качестве параметра принимает указатель, то при ее вызове необходимо передать адрес переменной: func(&a).

29 Что такое указатель?

Указатель — это переменная, которая хранит адрес области памяти. Объявление указателя: **тип *имя;**

Указатели бывают 3 типов: на объект, на функцию (**тип (*имя_фнкции) (список_параметров)**), на void.

Указатель на void применяется в тех случаях, когда тип объекта, адрес которого требуется хранить, не определён.

Указатели чаще всего используются при работе с динамической памятью.

Санкт-Петербургский государственный университет

Ответы

30 Что такое ссылка?

Ссылка — это, своего рода, альтернативное имя обекта. Ссылку можно рассматривать как указатель, который всегда разыменован. Объявление ссылки: **тип &имя**

Ссылка должна явно инициализироваться при объявлении. После инцилизации ссылке не может быть присвоена другая переменная. Ссылки чаще всего применяются в качестве параметров функций. Ссылка не занимает дополнительного места.

31 Что такое рефакторинг?

Рефакторинг — это изменение внутренней структуры программы, не затрагивающее её внешнего поведения и имеющее целью облегчить понимание её работы.

32 Что такое область видимости и какие бывают области видимости?

Область видимости — это часть программы, в пределах которой можно использовать объект. Как правило, область видимости ограничивается фигурными скобками. В зависимости от области видимости объекты могут быть глобальными или локальными.

33 Что такое пространство имён и что такое std?

Пространство имён — это область в рамках которой объявляются и определяются идентификаторы. Пространства имён используются для организации идентификаторов в логические группы и для избежания конфликта имён. Для объявления пространства имён используется ключевое слово namespace.

std — это пространство имён, которое содержит все имена из стандартной библиотеки C++.

Санкт-Петербургский государственный университет

Ответы

34 Как const влияет на переменные? Объясните что значит «const char* const comment»?

const указывает, что значение переменной не может быть изменено. Если const относится к типу переменной, то это означает, что данные не могут быть изменены. Если const относится к указателю, то это означает, что значение указателя не может быть изменено.

const char* const comment означает, что comment — это неизменяемый указатель на неизменяемые данные.

35 Как static влияет на глобальные и локальные переменные?

Локальная статическая переменная static int MAX = 100; будет инициализирована только один раз при первом входе в соответствующую область видимости и будет жить до конца выполнения программы. Но область видимости у неё будет в пределах её блока объявления.

Имена статических глобальных переменных известны только в файле, в котором они объвялены. Это позволяет использовать одну переменную в нескольких функциях этого файла без боязни получить непредвиденные изменения переменной из других файлов.

36 Какие основные типы данных есть в C++ и сколько они занимают памяти?

Оснвные типы C++: char, short, int, long, float, double, long double, void.

Спецификация C++ не определяет точных размеров для этих типов, а задаёт только отношение:
sizeof (char) ≤ sizeof (short) ≤ sizeof (int) ≤ sizeof (long)
sizeof (float) ≤ sizeof (double) ≤ sizeof (long double)

Для 64-битных машин эти типы обычно занимают памяти: char 1 байт, short 2 байта, int 4 байта, long 8 байт, float 4 байта, double 8 байт, long double 10 байт.

Санкт-Петербургский государственный университет

Ответы

37 Чем отличается префиксный инкремент (декремент) от постфиксного?

Префиксный инкремент (декремент) вначале меняют значение переменной, а потом её отдают в выражение.

Постфиксный — сначала отдают значение переменной в выражение, а уже потом её поменяют. Т.е. в выражении будет использовано значение переменной до применения над ней постфиксного инкремента (декремента).

38 Что такое разрядность машины?

Разрядность — это способность процессора осуществлять обработку и выполнение команд в определённом режиме битности. Разрядность определяет размер обработки данных за один такт. Современные компьютеры являются 64-х разрядными. Первый 64-х разрядный компьютер AMD Athlon 64 был создан в 2003 году.

39 Сколько места в памяти отводится под хранение указателя?

Размер указателя обычно равен разрядности машины, т.е. для 32-ых машин указатель будет занимать 4 байта, а для 64-ых — 8 байт.

40 Что означает тип указателя?

Тип указателя показывает сколько байт (ячеек памяти) надо взять, чтобы получить нужное значение. Это надо т.к. сам указатель указывает на адрес первого байта (ячейки) данных.

Санкт-Петербургский государственный университет

Ответы

37 Чем отличается префиксный инкремент (декремент) от постфиксного?

Префиксный инкремент (декремент) вначале меняют значение переменной, а потом её отдают в выражение.

Постфиксный — сначала отдают значение переменной в выражение, а уже потом её поменяют. Т.е. в выражении будет использовано значение переменной до применения над ней постфиксного инкремента (декремента).

38 Что такое разрядность машины?

Разрядность — это способность процессора осуществлять обработку и выполнение команд в определённом режиме битности. Разрядность определяет размер обработки данных за один такт. Современные компьютеры являются 64-х разрядными. Первый 64-х разрядный компьютер AMD Athlon 64 был создан в 2003 году.

39 Сколько места в памяти отводится под хранение указателя?

Размер указателя обычно равен разрядности машины, т.е. для 32-ых машин указатель будет занимать 4 байта, а для 64-ых — 8 байт.

40 Что означает тип указателя?

Тип указателя показывает сколько байт (ячеек памяти) надо взять, чтобы получить нужное значение. Это надо т.к. сам указатель указывает на адрес первого байта (ячейки) данных.

Санкт-Петербургский государственный университет

Ответы

41 Где хранятся переменные программы?

Переменные программы хранятся в некоторой области оперативной памяти, которая выделена под процесс запуска программы. Модель этой памяти делят на 2 больших части (в реальности её строение сложнее): стек и кучу. Стек используется для хранения переменных, размер которых можно вычислить во время компиляции. Куча (динамическая память) используется для тех переменных, размер которых становится известен только во время выполнения. Память для таких переменных выделяется оператором new (есть ещё несколько операторов выделения памяти, но их изучим позже).

42 Что такое стек (память)?

Стек — это часть модели оперативной памяти, которая выделена ОС под процесс запуска программы. Она хранит переменные, размер которых можно вычислить на момент компиляции. Особенность работы этой части памяти заключается в том, что переменные в неё попадают и удаляются из неё по принципу «последний вошёл и первый вышел» (LIFO). Т.е. работает как стек. Переменные удаляются из стека, когда заканчивается их область видимости. Взятие значений переменных в стеке очень быстрое, т.к. они расположены в верхушке стека и последовательно согласно использованию в коде.

43 Что такое куча (память)?

Куча — это часть модели оперативной памяти, которая выделена ОС под процесс запуска программы. Она хранит переменные, размер которых можно вычислить только во время выполнения. Как правило, память под такие переменные выделяется оператором new (есть и другие). Куча в отличие от стека работает как склад, на котором всё разбросано. Времени на переход между расположением переменных тратится много. Переменные, расположенные в куче, надо удалять вручную оператором delete или delete []. В C и C++, в отличие от java, java script, c#, python, не существует автоматического сборщика мусора.

Санкт-Петербургский государственный университет

Ответы

44 Что такое массив и чем он отличается от вектора?

Массив — это не структура данных, а представление последовательной области памяти. А вектор — это структура данных, которая является моделью динамического массива. Вектор — это класс, который реализует большой набор функциональности над массивом, например: добавление и удаление элементов, хранит размер последовательности, является итерируемым и т.д.

45 Чем отличается массив от указателя на массив?

Массив — это не структура данных, а представление последовательной области памяти.

Размер массива известен на этапе компиляции: int arr[5] = {1, 2, 3, 4, 5}. У массива можно узнать размер с помощью оператора sizeof: sizeof (arr) / sizeof (arr[0]).

Важно не путать массив с указателем на массив. Указатель на массив используется, когда размер последовательной области памяти для однотипных данных можно узнать только во время выполнения. Для выделения памяти под такие данные используют оператор new: int* arr_ptr = new int[n]. Узнать размер такого «массива» с помощью sizeof нельзя.

При передаче массива в функцию происходит его «преобразование» в указатель на массив. Поэтому в функцию всегда надо передавать размер массива дополнительной переменной.

46 Как найти размер массива?

У массива можно узнать размер с помощью оператора sizeof: sizeof (arr) / sizeof (arr[0]).

Но надо помнить, что при передаче массива в функцию происходит его «преобразование» в указатель на массив. Поэтому в функцию всегда надо передавать размер массива дополнительной переменной.

Санкт-Петербургский государственный университет

Ответы

47 Чем отличаются delete и delete []?

Оператор delete удаляет объект по указанному адресу. А оператор delete [] знает, что начиная с указанного адреса лежит последовательность объектов, которые ему надо удалить. Количество требуемых для удаления объектов он получает из того места, куда это количество для данной переменной записал оператор new []. Если вместо delete [] вызвать delete, то у программы будет неопределённое поведение: или падение, или утечка памяти.

48 Что будет, если забыть вызвать delete? Когда освободится та память?

Если забыть вызвать delete, то это приведёт к неопределённому поведению программы: её падению или утечке памяти. Вся память, в том числе и не очищенная из-под динамических переменных, будет очищенна при завершении программы. Это произойдёт т.к. ОС завершит процесс и освободит, выделенную под него область операционной памяти.

49 Как массив передаётся в функцию?

Массив в функцию передаётся как указатель на его первый элемент. Узнать размер массива в этом случае с помощью оператора sizeof нельзя. И приходится всегда вместе с массивом в функцию передавать дополнительной переменной и его размер.

50 Как происходит взятие элемента массива по индексу?

Взятие i-го (под i сейчас подразумевается не индекс) элемента массива происходит посредством перемещения указателя с первого элемента на i-1 позицию вправо и затем его разыменование. Т.е. arr[i — не индекс, а номер элемента] = *(arr + i -1). Именно поэтому индексация массива начинается с 0. Чтобы взять 2-ой элемент массива надо указать arr[1]. Т.к. сам arr уже указывает на свой 1-ый элемент, то чтобы взять 2-ой элемент указатель массива надо сдвинуть на 1 шаг вправо.