### Архитектура компьютера - 4 балла

1. Какое максимальное число можно сохранить в байте? Запишите его в двоичной системе. \*2 балла\*

Максимальное число, которое можно сохранить в байте, равно 255 (поскольку начинаем с 0). В двоичной системе это число записывается как 11111111.

2. Известно, что инструкции, которые видит процессор при выполнении программы, всего лишь числа.

Как процессор их различает? \*2 балла\*

Когда процессор выполняет программу, он читает специальные числа, которые говорят ему, какие действия нужно совершить. Каждая инструкция имеет свой уникальный номер, который говорит процессору, что делать.

Например, инструкция с номером 1 может означать "сложить два числа", и тд. Процессор знает, какую инструкцию выполнять, потому что он видит эти числа в программе.

Эти числа могут также содержать информацию о данных, с которыми нужно работать, и о том, где в памяти они находятся. Все это позволяет процессору выполнять разные действия в программе.

### Компилятор - 4 балла

1. Объясните, что делает компилятор C++ на высоком уровне. \*2 балла\*

Компилятор C++ на высоком уровне берет исходный код и преобразует его в исполняемую программу, которую можно запустить на компьютере. Этот процесс включает в себя анализ, проверку, оптимизацию и создание низкоуровневого кода, который понимает компьютер.

2. Объясните, что означает "предварительная обработка" на высоком уровне. Что делает препроцессор, когда видит `#include "file.h"`? \*2 балла\*

Предварительная обработка — это этап подготовки исходного кода перед компиляцией. Предварительная обработка готовит исходный код для компиляции, делая его более чистым для следующего этапа.

Когда препроцессор видит #include "file.h", он делает две вещи:

1. Идет ищет файл "file.h".
2. Вставляет содержимое этого файла в программу. Это позволяет использовать код из других файлов в программе.

### Программирование - 10 баллов

1. Что происходит в этом образце кода? Чему равны `a[0]` и `a[1]` в конце программы? \*5 баллов\*

```cpp

int main()

{

int a[2] = { };

a[0] = 10;

int\* p = &a[0];

\*p = \*p + 1;

p = p + 1;

\*p = \*p + 2;

return 0;

}

```

Создаётся массив a из двух элементов типа int. Каждый элемент массива равен 0, так как при инициализации фигурные скобки заполняют массив нулями.

Затем значение a[0] изменяется на 10.

Далее создаётся указатель p, который указывает на a[0].

\*p = \*p + 1 - увеличивает значение a[0] на 1. a[0] = 11

p = p + 1 - смещает указатель p на следующий элемент массива, теперь он указывает на a[1].

\*p = \*p + 2; увеличивает значение a[1] на 2. Теперь a[1] = 2.

В конце программы значения a[0] и a[1] равны 11 и 2.

2. Чему равны структуры в массиве в конце программы? Объясните свое рассуждение. \*5 баллов\*

```cpp

struct S

{

int values[2];

int\* p;

}

int main()

{

S arr[2] = { };

S\* sp = &arr[0];

int\* p = &sp->values[0];

S\* s1p = sp + 1;

s1p->p = p;

\*p = 10;

p = p + 1;

\*p = 20;

return 0;

}

```

Создается массив arr из 2 структур S. В начале значения values и указатель p в каждой структуре устанавливаются в нули.

Затем создается указатель sp, который указывает на первый элемент массива arr.

Создается указатель p, который указывает на первый элемент массива values - arr[0].values[0]

Создается указатель s1p, который указывает на второй элемент массива arr, то есть на arr[1].

s1p->p = p - присваивает указателю p во второй структуре (arr[1].p) значение указателя p, который указывал на первый элемент массива values в первой структуре (arr[0].values[0]).

\*p = 10 - устанавливает значение, на которое указывает p. Это означает, что arr[0].values[0] теперь равен 10.

p = p + 1 - смещает указатель p на следующий элемент массива, так что он теперь указывает на arr[0].values[1].

\*p = 20 - arr[0].values[1] равен 20.

В конце программы в первой структуре arr[0] массив values имеет значения arr[0].values[0] = 10 и arr[0].values[1] = 20, а во второй структуре arr[1] указатель p равен указателю p из первой структуры.

## Линкер - 6 баллов

Для компиляции кода использована команда `zig c++ main.cpp f.cpp`.

Объясните ошибку линкера, которая появится при попытке компиляции кода, если она есть.

Если код компилируется, сколько копий `f` окажется в итоговом исполняемом файле?

Что будет выведено при его запуске?

(Вы можете попробовать его запустить и интерпретировать результаты, это разрешено.)

1. \*3 балла\*

```cpp

// main.cpp

void f();

int main()

{

f();

return 0;

}

// f.cpp

void f();

```

Поскольку main.cpp вызывает функцию f(), линкер ожидает найти определение функции f() в одном из объектных файлов. В коде представлено объявление функции f(), но нет её определения. Это приведет к ошибке линкера.

Пример ошибки - undefined reference to `f()`

2. \*3 балла\*

```cpp

// main.cpp

#include <iostream>

inline void f()

{

std::cout << "Привет";

}

int main()

{

f();

return 0;

}

// f.cpp

#include <iostream>

void f()

{

std::cout << "Мир";

}

```

В коде функция f() определена дважды. Обе функции названы одинаково, и это приведет к конфликту имён при компиляции.

Пример ошибки

duplicate symbol 'void f()' in:

main.cpp.o

f.cpp.o

## Размеры - 4 балла

4. Каков размер типа данных (сколько байт он занимает в памяти) на 64-битной системе? \*4 балла каждый\*.

`int`

```cpp

struct S

{

int a;

int[5] b;

};

```

`void\*`

На сколько раз `size\_t` больше, чем `int\*`?

int: Размер типа int обычно составляет 4 байта (32 бита).

struct S: В этой структуре есть целое число int a, которое обычно составляет 4 байта. После него идет массив int[5] b, который содержит 5 целых чисел, по 4 байта на каждое, что составляет 20 байт. Таким образом, вся структура struct S будет занимать 24 байта.

void\*: Размер указателя void\* на 64-битной системе также обычно составляет 8 байт.

size\_t обычно также составляет 8 байт на

Размер size\_t в данном случае в два раза больше, чем int\*, потому что и тот, и другой занимают 8 байт.

## Выполнение функций и стек - 6 баллов

Опишите инструкции, которые выполнит этот код (вы можете создать таблицу, как в документе, или просто описать их словами).

- Не забудьте объяснить, как именно вызываются и возвращаются функции. \*2 балла\*

- Или нарисуйте, или объясните, как будет выглядеть стек на каждом этапе

(вы можете скопировать таблицу из предыдущего временного промежутка и изменить значения).

Включите \*адреса возврата\* в стек, но не беспокойтесь слишком о \*адресах памяти\*

(вы можете предположить, что адреса памяти имеют тот же размер, что и int, для простоты). \*4 балла\*

```cpp

void f()

{

int a;

int b;

a = 10;

b = a + 20;

// return;

}

void g()

{

f();

f();

// return;

}

```

Функция g() вызывается.

Внутри g(), функция f() вызывается первый раз.

int a - Создается целочисленная переменная a.

int b - Создается целочисленная переменная b.

a = 10 -Переменной a присваивается значение 10.

b = a + 20 -Переменной b присваивается значение 30.

Функция f() завершается и возвращается в g().

Внутри g(), функция f() вызывается второй раз и повторяет те же действия.

Функция g() завершается и программа завершает выполнение.

Перед тем как начать выполнение программы, стек пуст.

Вызывается функция g(). Стек будет выглядеть так:

|  |  |  |
| --- | --- | --- |
| Кард вызова | Локальные переменные | Адрес возврата |
| g() |  |  |

Функция g() вызывает f() первый раз

|  |  |  |
| --- | --- | --- |
| Кард вызова | Локальные переменные | Адрес возврата |
| f() | a,b | &g's return |
| g() |  |  |

Функция f() выполняется, создает локальные переменные a и b, и затем завершается, возвращаясь в функцию g().

Функция f() вызывается второй раз внутри функции g().

|  |  |  |
| --- | --- | --- |
| Кард вызова | Локальные переменные | Адрес возврата |
| f() | a,b | &g's return |
| f() | a,b | &g's return |
| g() |  |  |

Оба вызова f() создают свои собственные локальные переменные a и b. Когда каждый из них завершается, они возвращаются в функцию g().

Функция g() завершается, и программа заканчивает выполнение. Стек пуст.

Cтек вызовов функций был заполнен и опустошен в соответствии с порядком вызовов функций, и каждый кадр вызова содержит информацию о локальных переменных и адресе возврата.